# Uso del servicio TKG con el plano de control de IaaS de vSphere

Actualización 3

VMware vSphere 8.0
VMware vCenter 8.0
VMware ESXi 8.0



Puede encontrar la documentación técnica más actualizada en el sitio web de VMware by Broadcom en:

https://docs.vmware.com/es/

**VMware by Broadcom**
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

# Contenido

# Uso del servicio TKG con el plano de control de IaaS de vSphere

Esta documentación proporciona información para administrar el ciclo de vida de los clústeres de Tanzu Kubernetes Grid en el Supervisor en vSphere IaaS control plane.

Con Tanzu Kubernetes Grid en Supervisor, puede aprovisionar diferentes estilos de clústeres de carga de trabajo, incluidos los clústeres de Tanzu Kubernetes con valores predeterminados bien definidos y clústeres de ClusterClass con amplias opciones de definición. Al implementar Supervisor en zonas de vSphere, puede aprovisionar clústeres de TKG de alta disponibilidad que admitan cargas de trabajo con tolerancia a errores. El formato actualizado de la versión de Tanzu Kubernetes integra de forma nativa el repositorio de paquetes de Tanzu y admite varios sistemas operativos. Puede acceder y administrar clústeres de TKG en Supervisor mediante vCenter Single Sign-On y las Herramientas de la CLI de Kubernetes para vSphere, o mediante un proveedor de identidad externo y la CLI de Tanzu.

## Audiencia prevista

Esta documentación va dirigida a administradores de vSphere y operadores de Kubernetes que deseen aprovisionar y administrar el ciclo de vida de clústeres de TKG en el Supervisor en vSphere IaaS control plane.

## Versiones de vSphere y TKG aplicables

A menos que se indique lo contrario, esta documentación se aplica a la versión vSphere 8 de vSphere IaaS control plane y se actualiza para admitir TKG v3.x en Supervisor. Las actualizaciones de la documentación son acumulativas. La documentación archivada de las versiones anteriores se puede descargar en el sitio de documentación de VMware, en el encabezado **Paquetes de archivos**.

# Información actualizada

1

Esta documentación se actualiza periódicamente con información y correcciones nuevas según sea necesario.

En la tabla se muestra el historial de actualizaciones de esta documentación.

| Revisión | Descripción |
| --- | --- |
| 25 de junio de 2024 | Actualizaciones de la documentación para que sea compatible con vSphere 8 Update 3 y Servicio TKG 3.0, lo cual incluye:<br>■ Capítulo 3 Instalar y actualizar Servicio TKG<br>■ Capítulo 10 Ajuste de escala automático de clústeres de servicio TKG<br>■ Capítulo 20 Copia de seguridad y restauración de cargas de trabajo y clústeres de servicio TKG<br>■ Habilitar el adaptador Antrea-NSX para un clúster de servicio TKG<br>■ Instalar el servicio de proxy de administración de NSX<br>■ Capítulo 11 Instalar paquetes estándar en clústeres de Servicio TKG<br>■ Configurar MachineHealthCheck para clústeres v1beta1<br>■ API de clúster v1beta1 (actualizaciones)<br>  ■ controlPlaneCertificateRotation (actualizado)<br>  ■ podSecurityStandard (nuevo)<br>■ Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U2 y versiones posteriores) (actualizaciones)<br>■ Instalar y configurar el complemento de Velero para vSphere en un clúster de TKG (actualizaciones)<br>■ Escalar manualmente un clúster mediante Kubectl (actualizaciones) |
| 8 de marzo de 2024 | ■ Se actualizaron los requisitos de instantánea de CSI para indicar que las versiones mínimas son vSphere 8.0 Update 2+ con TKr v1.26 y versiones posteriores. Consulte Capítulo 15 Crear instantáneas en un clúster de Servicio TKG.<br>■ Se agregaron instrucciones para respaldar y restaurar cargas de trabajo de clústeres de TKGS mediante Velero con una instantánea de CSI. Consulte Copia de seguridad y restauración mediante Velero con instantánea de CSI.<br>■ Actualice los pasos de ejemplo y verificación para aprovisionar un clúster de API v1beta1 con un FQDN. Consulte Ejemplo de v1beta1: clúster con FQDN.<br>■ Se actualizó la descripción y la limitación de Kubernetes del permiso de función Puede Ver para espacios de nombres de vSphere. Consulte Enlaces y permisos de función.<br>■ Se agregaron instrucciones explícitas para instalar los certificados de CA raíz de confianza de vCenter en un host cliente Linux. Consulte Descargar los certificados de CA raíz de confianza para vCenter e instalarlos en un cliente Ubuntu. |

| Revisión | Descripción |
| --- | --- |
| 29 de febrero de 2024 | ■ Se actualizó la documentación a fin de admitir la versión para el público general de vSphere 8 U2 P03.<br>■ Se agregó documentación que respalda la creación de un clúster con un FQDN. Consulte API de clúster v1beta1 y Ejemplo de v1beta1: clúster con FQDN.<br>■ Se actualizaron los comandos de inicio de sesión de Supervisor para indicar que se puede utilizar el FQDN si está habilitado.<br>■ Se actualizó la documentación de PSA con ejemplos adicionales. Consulte Configurar PSA para TKR 1.25 y versiones posteriores.<br>■ Se agregaron instrucciones para instalar el operador de redes de NVIDIA en un clúster de carga de trabajo de TKGS a fin de ejecutar cargas de trabajo de vGPU. Consulte Flujo de trabajo de operadores de clúster para implementar cargas de trabajo de AI/ML en clústeres de servicio TKG.<br>■ Se agregó contenido para usar el asistente de creación de clases de máquina virtual, específicamente para crear una clase de máquina virtual personalizada correspondiente a las vGPU de NVIDIA Grip. Consulte Capítulo 13 Implementar cargas de trabajo de AI/ML en clústeres de Servicio TKG.<br>■ Se actualizó #unique_22 y Instalar y configurar el complemento de Velero para vSphere en un clúster de TKGcon vínculos a la matriz de compatibilidad de versiones para Velero. |
| 2 de febrero de 2024 | ■ Se actualizó la documentación de aprovisionamiento de clústeres de TKG para indicar que, en vSphere 8, la API v1alpha2 se convertirá automáticamente en la API v1alpha3. Consulte Acerca del aprovisionamiento de clústeres de TKG. |
| 31 de enero de 2024 | ■ Correcciones y modificaciones de errores varios. |
| 19 de enero de 2024 | ■ Se actualizó el tema para integrar un clúster de TKGS con un registro de contenedor privado. Consulte Integrar clústeres de Servicio TKG con un registro de contenedor privado. |
| 18 de enero de 2024 | ■ Se actualizó API de clúster v1beta1 a fin de aclarar que la variable `defaultRegistrySecret` está reservada para su uso con el registro de Harbor integrado. |
| 16 de enero de 2024 | ■ Se actualizó Capítulo 15 Crear instantáneas en un clúster de Servicio TKGpara indicar que debe utilizar el repositorio de paquetes de TKG Standard versión 2023.9.19 o posteriores. |
| 8 de enero de 2024 | ■ Actualizaciones menores varias. |
| 2 de enero de 2024 | ■ Actualizaciones menores varias. |
| 20 de diciembre de 2023 | ■ Se actualizó Uso de versiones de Kubernetes con clústeres de Servicio TKGcon el vínculo a la documentación para crear imágenes de TKr personalizadas.<br>■ Actualizaciones menores varias. |
| 6 de diciembre de 2023 | ■ Se actualizaron las Notas de la versión de TKR con las nuevas versiones de Tanzu Kubernetes.<br>■ Se actualizó Configurar PSA para TKR 1.25 y versiones posteriores para indicar que no se puede cambiar la PSA en los espacios de nombres del sistema.<br>■ Se actualizó Información sobre el modelo de actualización gradual para clústeres de Servicio TKG con consideraciones al utilizar varios grupos de nodos. |

| Revisión | Descripción |
| --- | --- |
| 21 de noviembre de 2023 | ■ Se actualizó Ejemplo de v1alpha3: TKC con red de pods enrutables y Ejemplo de v1beta1: clúster con red de pods enrutables para documentar que /21 es el tamaño mínimo de una red de pods enrutables.<br>■ Se actualizó Información sobre el modelo de actualización gradual para clústeres de Servicio TKG para aclarar cuándo se puede activar una actualización gradual para actualizaciones de Supervisor de vSphere 7.<br>■ Se actualizó Corregir errores de implementación de contenedorespara especificar cuándo se necesita PSP para TKr. |
| 13 de noviembre de 2023 | ■ Se actualizó la documentación para indicar que una clave de configuración válida debe constar de caracteres alfanuméricos, '-', '_' o '.' (p. ej., 'key.name', 'KEY_NAME' o 'key-name'). |
| 7 de noviembre de 2023 | ■ Se actualizó el tema para actualizar la cadena de versiones de TKr. Consulte Actualizar un clúster de TKG mediante el cambio de la versión de TKR. |
| 16 de octubre de 2023 | ■ Se corrigieron vínculos. |
| 13 de octubre de 2023 | ■ Se actualizaron las instrucciones de escalado de un clúster de TKG para incluir ejemplos cuando se utiliza la API v1beta1. Consulte Escalar manualmente un clúster mediante Kubectl. |
| 11 de octubre de 2023 | ■ Se actualizaron las instrucciones para conectarse a un clúster de TKG como usuario de OIDC mediante la CLI de Tanzu. Consulte Conectarse a un clúster de TKG como usuario de OIDC con la CLI de Tanzu. |
| 9 de octubre de 2023 | ■ Se actualizaron las instrucciones para asociar la biblioteca de contenido con espacios de nombres de vSphere. Consulte Asociar la biblioteca de contenido de TKR al Servicio TKG. |
| 3 de octubre de 2023 | ■ Se actualizó la documentación personalizada de ClusterClass de vSphere 8 U2 para incluir la anotación sobre cómo crear una variable ClusterClass personalizada no administrada. Consulte Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U2 y versiones posteriores). |
| 21 de septiembre de 2023 | Se actualizó la documentación de las funciones de la versión vSphere 8 U2, entre las que se incluyen:<br>■ Crear instantáneas de volumen para clústeres de TKG. Consulte Capítulo 15 Crear instantáneas en un clúster de Servicio TKG.<br>■ Instalar paquetes en clústeres de TKG. Consulte Capítulo 11 Instalar paquetes estándar en clústeres de Servicio TKG.<br>■ Aprovisionar clústeres de ClusterClass personalizados mediante la API v1beta1. Consulte Usar la API v1beta1 del clúster.<br>■ Consideraciones de la actualización gradual para clústeres de TKG. Consulte Información sobre el modelo de actualización gradual para clústeres de Servicio TKG.<br>■ Varias actualizaciones y ediciones de mantenimiento.<br>**Nota** Asegúrese de comprobar las notas de la versión para vSphere IaaS control plane y las versiones de Tanzu Kubernetes. |
| 16 de agosto de 2023 | ■ Actualizar los ejemplos de aprovisionamiento de clústeres de TKG para v1alpha3 TKC con pods enrutables y el clúster v1beta1 con ClusterClass personalizado. Consulte Capítulo 7 Aprovisionar clústeres del servicio de TKG. |

| Revisión | Descripción |
| --- | --- |
| 15 de agosto de 2023 | ■ Se actualizó la documentación de instalación del paquete de Tanzu. Consulte Capítulo 11 Instalar paquetes estándar en clústeres de Servicio TKG.<br>■ Correcciones y modificaciones menores. |
| 10 de agosto de 2023 | ■ Se agregó documentación para insertar imágenes de paquete de Tanzu del registro de Harbor público a un registro de Harbor privado que se ejecute como un servicio de Supervisor. Consulte Capítulo 14 Uso de registros privados con clústeres Servicio TKG.<br>■ Correcciones y modificaciones menores. |
| 7 de agosto de 2023 | ■ Se eliminó la documentación de ExternalTrafficPolicy y LoadBalancerSourceRanges para el tipo de servicio: LoadBalancer con NSX. Estas funciones no son compatibles. |
| 2 de agosto de 2023 | ■ Se agregó un tema sobre la conexión a clústeres de TKG mediante Tanzu CLI y kubectl. Consulte #unique_36.<br>■ Correcciones y modificaciones menores. |
| 31 de julio de 2023 | ■ Correcciones y modificaciones menores. |
| 27 de julio de 2023 | ■ Se actualizó la documentación para admitir TKG 2.2. |
| 21 de julio de 2023 | ■ Se agregó una lista de los permisos necesarios para administrar la biblioteca de contenido para las versiones de Tanzu Kubernetes. |
| 12 de julio de 2023 | ■ Se actualizó el nombre de las funciones de editor y visor que se editarán y verán. Consulte Acerca de la administración de identidades y acceso para clústeres de Servicio TKG. |
| 10 de julio de 2023 | ■ Se agregaron instrucciones para editar las bibliotecas de contenido que pueden utilizar los clústeres de TKG en Supervisor. Consulte Editar una biblioteca de contenido existente.<br>■ Se actualizó el tema para recopilar paquetes de soporte para clústeres de TKG en el Supervisor. Consulte Extraer registros para solucionar problemas de clústeres de TKG en Supervisor. |
| 7 de julio de 2023 | ■ Se actualizó el tema donde se explican los dos tipos de clústeres diferentes que se pueden aprovisionar con TKG 2.0 en el Supervisor. Consulte Acerca del aprovisionamiento de clústeres de TKG.<br>■ Se agregó un tema con consideraciones sobre el aprovisionamiento de clústeres con montajes de volúmenes de nodos personalizados. Consulte Consideraciones para usar montajes de volumen de nodo. |
| 3 de julio de 2023 | ■ Se actualizó el tema para crear una función y un grupo de operadores de vSphere IaaS control plane. Consulte Crear un grupo y una función dedicados para los operadores de plataforma. |
| 30 de junio de 2023 | ■ Se actualizó el script para instalar el administrador de certificados. Consulte #unique_41.<br>■ Se agregaron instrucciones para registrar una instancia de Tanzu Mission Control autoadministrado con el Supervisor. Consulte Registrar Tanzu Mission Control autoadministrado con Supervisor.<br>■ Se actualizó la instrucción del host de salto de máquina virtual para entornos de vDS. Consulte Crear una máquina virtual de host de salto de Linux. |

| Revisión | Descripción |
| --- | --- |
| 26 de junio de 2023 | ■ Se actualizó la lista de las API del clúster de TKG con detalles sobre las que son compatibles con el Supervisor de vSphere 8. Consulte Acerca del aprovisionamiento de clústeres de TKG.<br>■ Se actualizó la lista de permisos para crear una función de operadores de clúster de TKG dedicados. Consulte Crear un grupo y una función dedicados para los operadores de plataforma.<br>■ Se actualizó el comando para instalar Contour mediante la CLI de Tanzu. Consulte Crear el repositorio de paquetes. |
| 13 de junio de 2023 | ■ Se agregó un tema para crear una función y un grupo de operadores de clúster de TKG dedicados. Consulte Crear un grupo y una función dedicados para los operadores de plataforma.<br>■ Se actualizó el tema para crear un clúster v1beta1 mediante una red de pods enrutables. Consulte Ejemplo de v1beta1: clúster con red de pods enrutables.<br>■ Se eliminó el tema sobre cómo crear un clúster v1alpha3 de Tanzu Kubernetes con una red de pods enrutables debido a un problema conocido. Consulte las notas de la versión para obtener más información. |
| 9 de junio de 2023 | ■ Se agregó un tema para eliminar un espacio de nombres de vSphere del Supervisor con un requisito previo para eliminar primero todos los clústeres de TKG aprovisionados en ese espacio de nombres de vSphere. Consulte Quitar un espacio de nombres de vSphere. |
| 6 de junio de 2023 | ■ Se agregó el gráfico del mapa de documentación de vSphere IaaS control plane para desplazarse por las diversas publicaciones que componen el conjunto de documentos. Consulte Uso del servicio TKG con el plano de control de IaaS de vSphere. |
| 5 de junio de 2023 | ■ Se realizaron cambios editoriales en varios temas en las siguientes secciones: Capítulo 2 Ejecución de clústeres de Servicio TKG y Capítulo 9 Actualizar clústeres de servicio TKG.<br>■ Se actualizó Administrar certificados TLS para clústeres del servicio TKG con referencias a la Guía de certificados de vSphere with Tanzu y otros cambios. |
| 1 de junio de 2023 | ■ Se actualizaron los ejemplos de aprovisionamiento de clústeres. Consulte Usar la API v1alpha3 del clúster de Tanzu Kubernetes y Usar la API v1beta1 del clúster.<br>■ Se actualizaron los ejemplos de escalado de clústeres. Consulte Escalar manualmente un clúster mediante Kubectl. |
| 26 de mayo de 2023 | ■ Se actualizó el tema de actualización gradual para clústeres de TKG con detalles adicionales. Consulte Información sobre el modelo de actualización gradual para clústeres de Servicio TKG. |
| 23 de mayo de 2023 | ■ Se refactorizó la documentación para utilizar TKG 2.0 en lugar de TKG 2. |
| 22 de mayo de 2023 | ■ Se actualizó el comando para crear una directiva de seguridad de pods con espacio de nombres. Consulte Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG.<br>■ Se actualizó la documentación del flujo de trabajo de actualización. Consulte #unique_51. |
| 12 de mayo de 2023 | ■ Se agregó una nota que indica que solo se pueden aprovisionar nuevos clústeres de TKG 2 en una instancia de Supervisor que se implementa en tres zonas de vSphere. |
| 11 de mayo de 2023 | ■ Se agregó un tema sobre la expansión de volúmenes persistentes. Consulte Expansión de volúmenes persistentes para clústeres de Servicio TKG. |

| Revisión | Descripción |
|---|---|
| 9 de mayo de 2023 | ■ Se actualizó el gráfico de la topología con zonas para indicar que los pods se implementan en las zonas de este ejemplo. Consulte Arquitecturas de referencia para clústeres de Servicio TKG.<br>■ Se actualizó Uso de espacios de nombres de vSphere con clústeres de Servicio TKG con detalles de almacenamiento adicionales.<br>■ Se actualizó Capítulo 8 Operar clústeres del servicio de TKG con detalles adicionales sobre la supervisión del almacenamiento persistente. |
| 4 de mayo de 2023 | ■ Se actualizó Crear el repositorio de paquetes para utilizar `--package`.<br>■ Se actualizó Configurar PSP para TKR 1.24 y versiones anteriores para aclarar que los PSP siguen siendo necesarios para los clústeres de TKG 2 y que no se admite la controladora de admisión de seguridad de pods.<br>■ Se actualizó #unique_22. |
| 1 de mayo de 2023 | ■ Se fijó el vínculo. |
| 4 de abril de 2023 | ■ Se realizaron actualizaciones y mejoras generales para la versión de vSphere 8 Update 1.<br>■ Se actualizó Uso de versiones de Kubernetes con clústeres de Servicio TKG para la versión vSphere 8 U1, incluida la comprobación de la compatibilidad de TKR con Supervisor y con TKG 2, la eliminación del sufijo `-zshippable` de la cadena TKR NAME y la creación de su propia TKR.<br>■ Se actualizó la documentación de especificación de API de clúster v1beta1 .<br>■ Se actualizó Integrar clústeres de Servicio TKG con un registro de contenedor privado con ejemplos.<br>■ Se actualizó Configurar un editor de texto para Kubectl con ejemplos adicionales de Windows.<br>■ Se actualizó Realizar una copia de seguridad y restaurar cargas de trabajo de clústeres de TKG en Supervisor mediante Restic y Velero independientes con correcciones de errores. |

# Ejecución de clústeres de Servicio TKG

2

En esta sección se describen los componentes y se enumeran los requisitos para ejecutar clústeres de Servicio TKG.

Lea los siguientes temas a continuación:

- Componentes de Servicio TKG
- Implementar clústeres de Servicio TKG
- Arquitecturas de referencia para clústeres de Servicio TKG

## Componentes de Servicio TKG

Servicio TKG proporciona administración del ciclo de vida de autoservicio de los clústeres de carga de trabajo de Kubernetes. El servicio TKG con Supervisor está optimizado para el entorno de vSphere y se integra con la infraestructura subyacente, incluidos vCenter, ESXi, redes virtuales y almacenamiento nativo en la nube. Con el servicio TKG, puede aprovisionar clústeres de Kubernetes compatibles y mantener la simultaneidad ascendente.

El Servicio TKG incluye varios componentes integrados con el vSphere IaaS control plane.

Figura 2-1. Componentes de Servicio TKG

Espacios de nombres de usuario
CPI
CSI
CNI
Auth
Pod
Pod
Pod
Pod
Clúster de TKG
vm
Nodo
vm
Nodo
vm
Nodo
Recurso de TKG
Recurso del clúster
Recursos de máquina virtual
Recursos del supervisor
Espacios de nombres del sistema
Recurso de TKG
Controladores de la API del clúster
Proveedor de API del clúster
Servicio de máquina virtual
vCenter Server
Integración de la interfaz de usuario
Espacio de nombres de vSphere
Control de acceso basado en funciones
Biblioteca de contenido de TKR
Clases de máquina virtual
Almacenamiento nativo en la nube (CNS)
Clúster de vSphere
Zona de vSphere
vSphere Distributed Switch
Supervisor
NCP-CNI
CNS-CSI

## Administración de cargas de trabajo

Administración de carga de trabajo es una solución de VMware que proporciona un plano de control de Kubernetes en una infraestructura de vSphere nativa. Al habilitar la administración de cargas de trabajo, se permite implementar uno o varios Supervisores en el entorno de vSphere. La administración de cargas de trabajo se incluye en un paquete con vCenter Server, pero su licencia se adquiere por separado.

## Supervisor

El Supervisor es un clúster de Kubernetes que funciona como el plano de control para administrar los clústeres de carga de trabajo. Configure Supervisor para admitir desarrolladores que aprovisionen y operen clústeres de Tanzu Kubernetes Grid.

## Implementación de un Supervisor: clúster de vSphere

La forma tradicional de implementar un Supervisor es en un único clúster de vSphere. Un clúster de vSphere es una recopilación de hosts ESXi administrados por una instancia de vCenter Server. Es necesario configurar un clúster de vSphere con características específicas para admitir el Supervisor. Algunas de estas son:

- Varios hosts ESXi conectados mediante un conmutador vSphere Distributed Switch (VDS)
- El almacenamiento compartido está configurado, como vSAN o NFS
- vSphere HA y DRS totalmente automatizado están habilitados
- Lifecycle Manager está habilitado

- Las redes están configuradas, ya sea NSX con su equilibrador de carga integrado, o VDS con un equilibrador de carga externo

Desde el punto de vista arquitectónico, para las implementaciones de producción de TKG en Supervisor, debe separar el clúster o el host del plano de administración donde se ejecuta vCenter Server del clúster del plano informático donde se habilitará Supervisor. Si utiliza un clúster de vSphere para alojar vCenter Server, este clúster no debe tener habilitado DRS. Consulte la documentación de vCenter para obtener detalles.

## Implementación de un Supervisor: zonas de vSphere

vSphere 8 presenta las zonas de vSphere. Puede asignar clústeres de vSphere a zonas de vSphere para proporcionar alta disponibilidad y tolerancia a errores para Supervisor. La implementación de Supervisor en las zonas de vSphere le permite aprovisionar clústeres de TKG en zonas de disponibilidad específicas.

En la implementación de un Supervisor en un solo clúster de vSphere, se establece una relación uno a uno entre el Supervisor y el clúster de vSphere. En una implementación de Supervisor en zonas, Supervisor se extiende a los clústeres de vSphere para proporcionar alta disponibilidad y dominios de errores a los clústeres de TKG en Supervisor.

Un administrador de vSphere crea las tres zonas de vSphere y las asocia con un clúster de vSphere. Una implementación del Supervisor en las tres zonas de vSphere tiene requisitos especiales además de los requisitos del clúster de vSphere, entre los que se incluyen:

- Tres zonas de vSphere conectadas mediante un conmutador vSphere Distributed Switch (VDS)
- Cada zona de vSphere contiene un único clúster de vSphere
- Un Supervisor debe implementarse exactamente en 3 zonas de vSphere
- Una directiva de almacenamiento de vSphere está disponible en la zona de vSphere

## espacio de nombres de vSphere

Un espacio de nombres de vSphere es un espacio de nombres en un Supervisor donde se aprovisionan uno o varios clústeres de Tanzu Kubernetes Grid. Para cada espacio de nombres de vSphere, configure el control de acceso basado en funciones, el almacenamiento persistente, los límites de recursos, la biblioteca de imágenes y las clases de máquinas virtuales.

## Servicio TKG

Servicio TKG es una implementación del proyecto de API de clúster de código abierto que define un conjunto de controladoras y recursos personalizados para administrar el ciclo de vida de los clústeres de Kubernetes. Tanzu Kubernetes Grid es un componente de Supervisor.

Servicio TKG tiene tres capas de controladoras para administrar el ciclo de vida de los clústeres de TKG, incluidos servicio de máquina virtual, la API del clúster y el complemento de proveedor de nube.

**VM Operator**

La controladora del servicio de máquina virtual proporciona una API declarativa estilo Kubernetes para la administración de las máquinas virtuales y los recursos de vSphere asociados. El servicio de máquina virtual presenta el concepto de una clase de máquina virtual que representa una configuración de hardware reutilizable y abstracta. Servicio TKG usa el servicio de máquina virtual para administrar el ciclo de vida de las máquinas virtuales de plano de control y de nodo de trabajo que alojan un clúster de carga de trabajo.

**API del clúster**

La controladora de la API del clúster proporciona API de estilo Kubernetes declarativas para la creación, la configuración y la administración de clústeres. Las entradas de la API del clúster incluyen un recurso que describe el clúster, un conjunto de recursos que describen las máquinas virtuales que componen el clúster y un conjunto de recursos que describen los complementos del clúster.

**Complemento de proveedor de nube**

Servicio TKG aprovisiona clústeres de carga de trabajo que incluyen los componentes necesarios para integrarse con los recursos de espacio de nombres de vSphere subyacentes. Estos componentes incluyen un complemento de proveedor de nube que se integra con la instancia de Supervisor. TKG utiliza el complemento de proveedor de nube para enviar solicitudes de volúmenes persistentes al Supervisor que se integra con el almacenamiento nativo en la nube (Cloud Native Storage, CNS) de VMware.

## Versiones de Tanzu Kubernetes

Una versión de Tanzu Kubernetes proporciona la distribución de software de Kubernetes y los complementos firmados y compatibles con VMware para su uso con clústeres de Tanzu Kubernetes Grid.

Cada versión de Tanzu Kubernetes se distribuye como una plantilla de máquina virtual (archivo OVA). El Tanzu Kubernetes Grid utiliza la plantilla de OVA para construir los nodos de máquina virtual de los clústeres de TKG. Las versiones de Tanzu Kubernetes se determinan de acuerdo con el control de versiones de Kubernetes e incluyen personalizaciones y optimizaciones del sistema operativo para la infraestructura de vSphere.

Para obtener una lista de las versiones de Tanzu Kubernetes y la compatibilidad con el Supervisor, consulte las Notas de la versión de las versiones de Tanzu Kubernetes. Consulte también la vSphere IaaS control planeDirectiva de soporte técnico.

**Biblioteca de contenido**

Las versiones de Tanzu Kubernetes se ponen a disposición de los clústeres de TKG mediante una biblioteca de contenido de vCenter. Puede crear una biblioteca de contenido suscrita y recibir automáticamente las TKR cuando estén disponibles mediante VMware, o bien utilizar una biblioteca de contenido local y cargar las TKR manualmente.

## Componentes del clúster de Servicio TKG

Los componentes que se ejecutan en un clúster de Servicio TKG proporcionan funcionalidad para cuatro áreas del producto: autenticación, almacenamiento, redes y equilibrio de carga.

**Webhook de autenticación**

El webhook de autenticación se ejecuta como pod dentro del clúster para validar los tokens de autenticación de usuario.

Los clústeres de TKG admiten dos formas de autenticación: uso de vCenter Single Sign-On y uso de un proveedor de identidad externo que admite el protocolo de Open ID Connect (OIDC).

TKG ejecuta el cliente OIDC de Pinniped en el Supervisor y en los nodos de clúster de TKG. Configurar un proveedor de OIDC externo para Supervisor configura automáticamente los componentes de Pinniped.

TKG admite varios clientes para la autenticación con el Supervisor, incluidos el complemento de vSphere para kubectl y la CLI de Tanzu.

**Interfaz de almacenamiento de contenedor (Container Storage Interface, CSI)**

Un complemento de CSI paravirtual es un pod de Kubernetes que se ejecuta dentro de un clúster de TKG y se integra con el almacenamiento nativo en la nube (Cloud Native Storage, CNS) de VMware a través de Supervisor. Un pod de Kubernetes que se ejecuta dentro de un clúster de TKG puede montar tres tipos de discos virtuales: efímero, volumen persistente e imagen de contenedor.

**Almacenamiento transitorio**

Un pod requiere almacenamiento transitorio para almacenar datos efímeros, por ejemplo, objetos de Kubernetes como registros, volúmenes y mapas de configuración. El almacenamiento transitorio dura mientras existe el pod. Los datos efímeros persisten los reinicios del contenedor, pero una vez que el pod se elimine, los datos efímeros del almacenamiento del disco virtual desaparecen.

**Almacenamiento persistente**

TKG aprovecha el marco de directivas de almacenamiento de vSphere para definir clases de almacenamiento y reservar volúmenes persistentes. Los clústeres de TKG envían solicitudes de volúmenes persistentes al Supervisor que se integra con el almacenamiento nativo en la nube (Cloud Native Storage, CNS) de VMware. Los volúmenes persistentes se aprovisionan en los nodos del clúster dinámicamente usando la clase de almacenamiento, o de forma manual.

### Almacenamiento de imágenes de contenedor

Los contenedores dentro de un pod de Kubernetes utilizan imágenes que incluyen el software que se ejecutará. El pod monta imágenes utilizadas por sus contenedores como discos virtuales de imagen. Cuando el pod completa su ciclo de vida, los discos virtuales de imagen se desasocian del pod. Kubelet es responsable de extraer las imágenes de contenedor del registro de imágenes y transformarlas en discos virtuales que se ejecuten dentro del pod.

## Interfaz de red de contenedor (Container Network Interface, CNI)

El complemento de la interfaz de red de contenedor es un complemento de CNI que proporciona redes de pod.

Los clústeres de TKG admiten las siguientes opciones de interfaz de red de contenedor: Antrea (predeterminada) y Calico. Además, TKG proporciona la CNI enrutada de NSX Antrea para implementar redes de pods enrutables.

En la tabla se resumen las funciones de redes de clústeres de TKG y su implementación.

**Tabla 2-1. Redes de clústeres de Servicio TKG**

| Extremo | Proveedor | Descripción |
| --- | --- | --- |
| Conectividad de pods | Antrea o Calico | Interfaz de red de contenedor para pods. Antrea utiliza Open vSwitch. Calico utiliza el puente de Linux con BGP. |
| Tipo de servicio: ClusterIP | Antrea o Calico | Tipo de servicio de Kubernetes predeterminado al que solo se puede acceder en el clúster. |
| Tipo de servicio: NodePort | Antrea o Calico | Permite el acceso externo a través de un puerto abierto en cada nodo de trabajo mediante el proxy de red de Kubernetes. |
| Directiva de red | Antrea o Calico | Controla el tráfico que se permite hacia y desde los pods seleccionados y los endpoints de red. Antrea utiliza Open vSwitch. Calico utiliza tablas de IP de Linux. |

## Implementación de proveedores de nube

La implementación de proveedor de nube admite la creación de servicios de entrada y de equilibrador de carga de Kubernetes.

Tabla 2-2. Equilibrio de carga de TKG

| Extremo | Proveedor | Descripción |
| --- | --- | --- |
| Tipo de servicio: LoadBalancer | El equilibrador de carga integrado de NSX (parte de la pila de red de NSX)<br>NSX Advanced Load Balancer (instalación independiente para su uso con redes VDS)<br>HAProxy (instalación independiente para su uso con redes VDS) | Para el equilibrador de carga integrado de NSX y un servidor virtual por definición de tipo de servicio.<br>Para NSX Advanced Load Balancer, consulte esa sección en esta documentación.<br>Para HAProxy, consulte esa sección de esta documentación.<br>**Nota** Es posible que algunas características de equilibrio de carga no se encuentren disponibles en cada tipo de equilibrador de carga compatible, por ejemplo, las direcciones IP estáticas. |
| Entrada de clúster | Controladora de entrada de terceros | Proporciona enrutamiento para el tráfico de pod entrante. Puede utilizar cualquier controladora de entrada de terceros, como Contour. |

## API del clúster de Servicio TKG

Servicio TKG proporciona dos API para aprovisionar y administrar el ciclo de vida de los clústeres de TKG.

- Versión de API `v1alpha3` para clústeres de Tanzu Kubernetes
- Versión de API `v1beta1` para clústeres en función de una ClusterClass

La API v1alpha3 permite crear clústeres de Kubernetes conformes del tipo `TanzuKubernetesCluster`. Este tipo de clúster se configura previamente con valores predeterminados comunes para el aprovisionamiento rápido y se puede personalizar. La API v1beta1 permite crear clústeres de Kubernetes conformes de tipo `Cluster` basados en una `ClusterClass` predeterminada que proporciona VMware.

**Nota** Para actualizar vSphere IaaS control plane de vSphere 7 a vSphere 8, el clúster de TKG debe ejecutar la API v1alpha2. La API v1alpha2 se introdujo con v7.0 Update 3. La API v1alpha1 está obsoleta. Para obtener más información, consulte #unique_51.

## Clientes de clúster de Servicio TKG

TKG en el Supervisor de vSphere 8 admite varias interfaces de cliente para el aprovisionamiento, la supervisión y la administración de los clústeres de TKG.

- vSphere Client para configurar el Supervisor y supervisar los clústeres de TKG implementados.

- complemento de vSphere para kubectl para la autenticación en el Supervisor y los clústeres de TKG mediante vCenter Single Sign-On.
- kubectl para aprovisionar y administrar el ciclo de vida de los clústeres de TKG de forma declarativa e interactuar con el Supervisor.
- complemento auxiliar de credenciales de vSphere Docker para insertar y extraer imágenes hacia y desde un registro de contenedor.
- CLI de Tanzu para aprovisionar clústeres mediante comandos y para instalar paquetes de Tanzu.
- Interfaz web de Tanzu Mission Control para administrar los clústeres de TKG.

# Implementar clústeres de Servicio TKG

Para implementar clústeres de Servicio TKG, instale o actualice Servicio TKG, utilice una biblioteca de contenido para almacenar las versiones de Tanzu Kubernetes y configure uno o varios espacios de nombres de vSphere para alojar clústeres de TKG.

## Implementar clústeres de servicio TKG

Para implementar clústeres de Servicio TKG, complete el proceso de habilitación de **Administración de cargas de trabajo** y configure una instancia de Supervisor.

**Nota** Consulte *Instalar y configurar el plano de control de IaaS de vSphere* para obtener instrucciones.

Después de habilitar la **Administración de cargas de trabajo** e implementar Supervisor, complete las siguientes tareas como preparación para el aprovisionamiento de clústeres de Servicio TKG.

| Tarea | Descripción |
| --- | --- |
| Instalar o actualizar Servicio TKG | Consulte Capítulo 3 Instalar y actualizar Servicio TKG. |
| Instalar las CLI y conectarse al Supervisor | Consulte Capítulo 4 Configurar identidad y acceso de los clústeres de Servicio TKG. |
| Crear y sincronizar la biblioteca de contenido para las versiones de Tanzu Kubernetes (TKr) | La biblioteca de contenido integra el entorno de TKG con la red de entrega de contenido de VMware a través de la cual se distribuyen las TKr. Puede utilizar una biblioteca de contenido local o suscrita en función de sus requisitos.<br>Las TKr existentes deben actualizarse para aprovechar las nuevas funciones de TKG. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG. |

| Tarea | Descripción |
| --- | --- |
| Crear y configurar espacios de nombres de vSphere para alojar clústeres de servicio TKG | Al definir un espacio de nombres de vSphere, se crea el segmento de red para alojar los clústeres de servicio TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.<br>Es necesario configurar cada espacio de nombres de vSphere con:<br>■ Usuarios y funciones del clúster<br>■ Biblioteca de contenido TKr sincronizada<br>■ Clases de máquina virtual asociadas<br>■ Directiva de almacenamiento de vSphere para nodos de clúster de servicio TKG y volúmenes persistentes<br>Si utiliza un supervisor implementado en tres zonas de vSphere, la directiva de almacenamiento de vSphere debe especificar la topología `Zonal`. Consulte Crear una directiva de almacenamiento de vSphere para clústeres de Servicio TKG. |
| Aprovisionar clústeres de Servicio TKG | Consulte Capítulo 7 Aprovisionar clústeres del servicio de TKG. |
| Operar clústeres de Servicio TKG | Consulte Capítulo 8 Operar clústeres del servicio de TKG. |
| Implementar cargas de trabajo de clústeres | Consulte Capítulo 12 Implementar cargas de trabajo en clústeres de Servicio TKG. |
| Mantener clústeres de Servicio TKG | Consulte Capítulo 9 Actualizar clústeres de servicio TKG. |
| Solucionar problemas de clústeres de Servicio TKG | Consulte Capítulo 21 Solucionar problemas de clústeres de servicio TKG. |

# Arquitecturas de referencia para clústeres de Servicio TKG

Este tema proporciona un conjunto de arquitecturas de referencia para clústeres de Servicio TKG con diversas topologías de implementación.

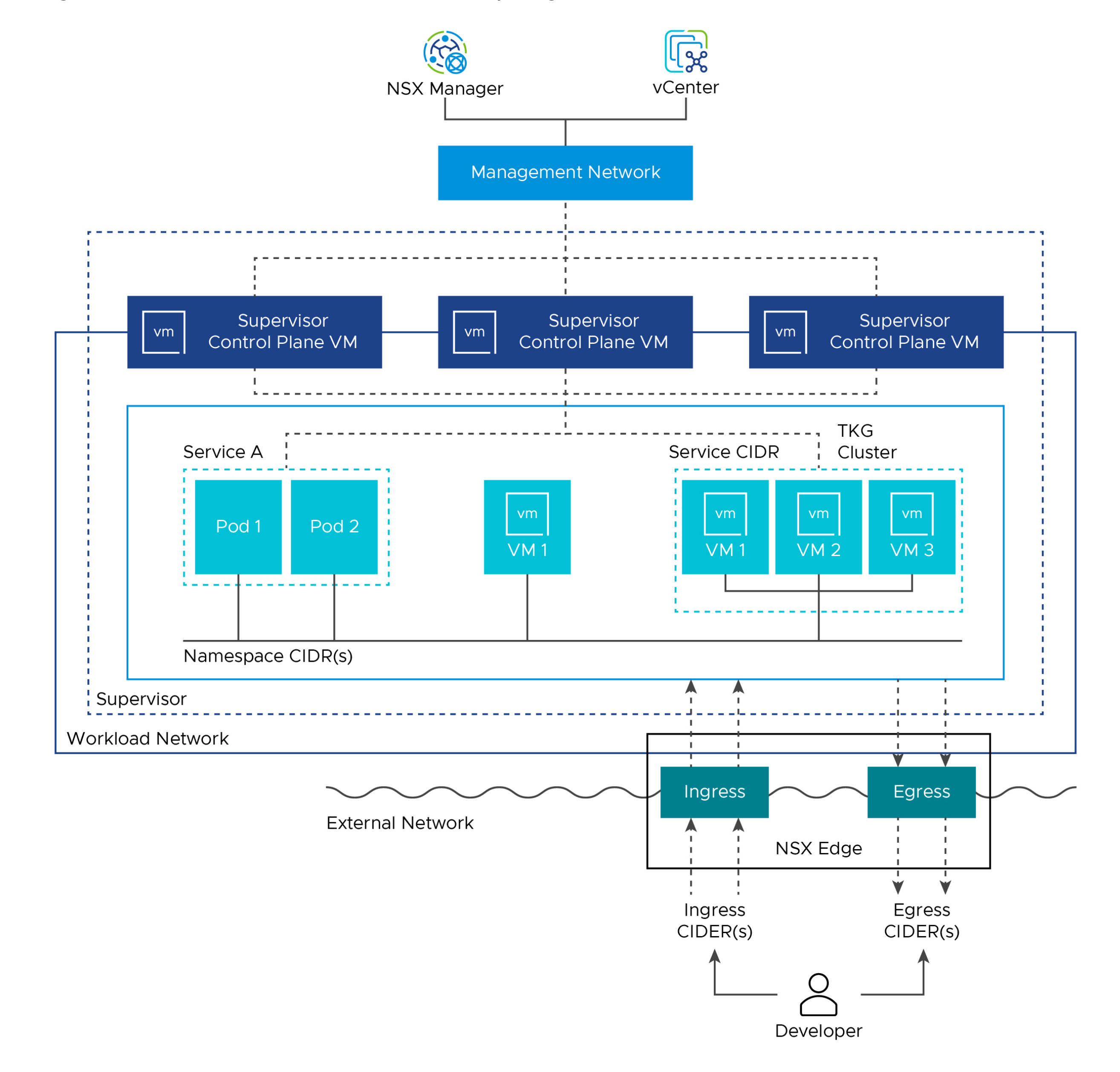
## Clúster de Servicio TKG con topología de NSX

La arquitectura de referencia ilustra un clúster de TKG en el Supervisor con redes de NSX. En un entorno de este tipo, el plano de administración aloja vCenter Server e instancias de NSX Manager. El plano informático aloja nodos de NSX Edge y nodos de Supervisor.

Están presentes los siguientes objetos de redes NSX:

- Puerta de enlace de nivel 1 (enrutador)
- Segmento (conmutador) vinculado a la puerta de enlace
- Servidor del equilibrador de carga
- Grupo de servidores para cada servidor virtual del plano de control del clúster de TKG
- Servidor virtual para cada instancia del equilibrador de carga del servicio de Kubernetes

Figura 2-2. Clúster de Servicio TKG con topología de NSX

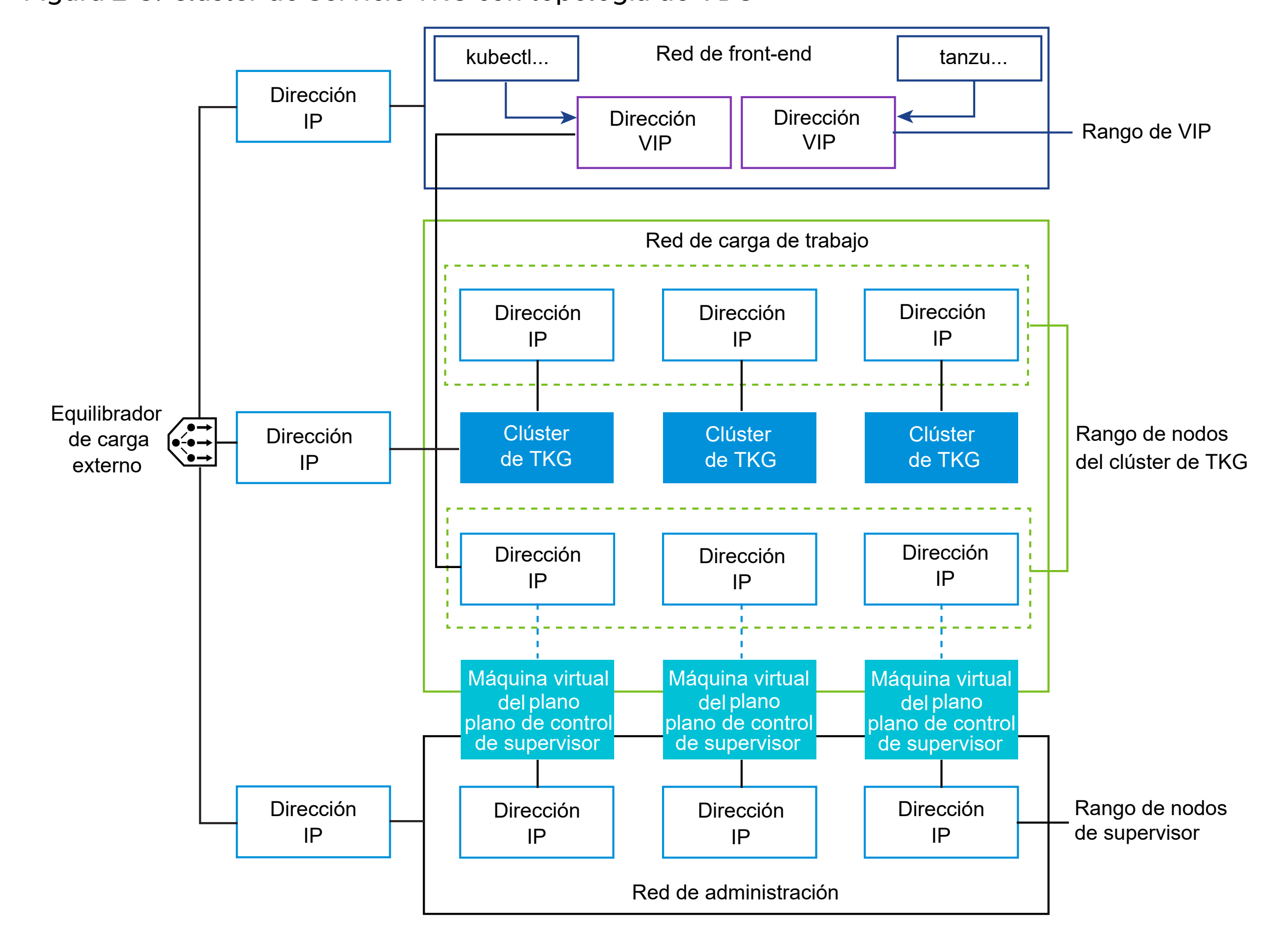



## Clúster de Servicio TKG con topología de VDS

La arquitectura de referencia ilustra un clúster de Servicio TKG con redes de VDS y un equilibrador de carga externo. En un entorno de este tipo están presentes las siguientes redes:

- Red de administración para máquinas virtuales de plano de control del Supervisor
- Red de carga de trabajo para clústeres de TKG
- Red de front-end a través de la cual los desarrolladores se conectan a clústeres de Servicio TKG

Figura 2-3. Clúster de Servicio TKG con topología de VDS




## Clúster de Servicio TKG con topología de zonas de vSphere

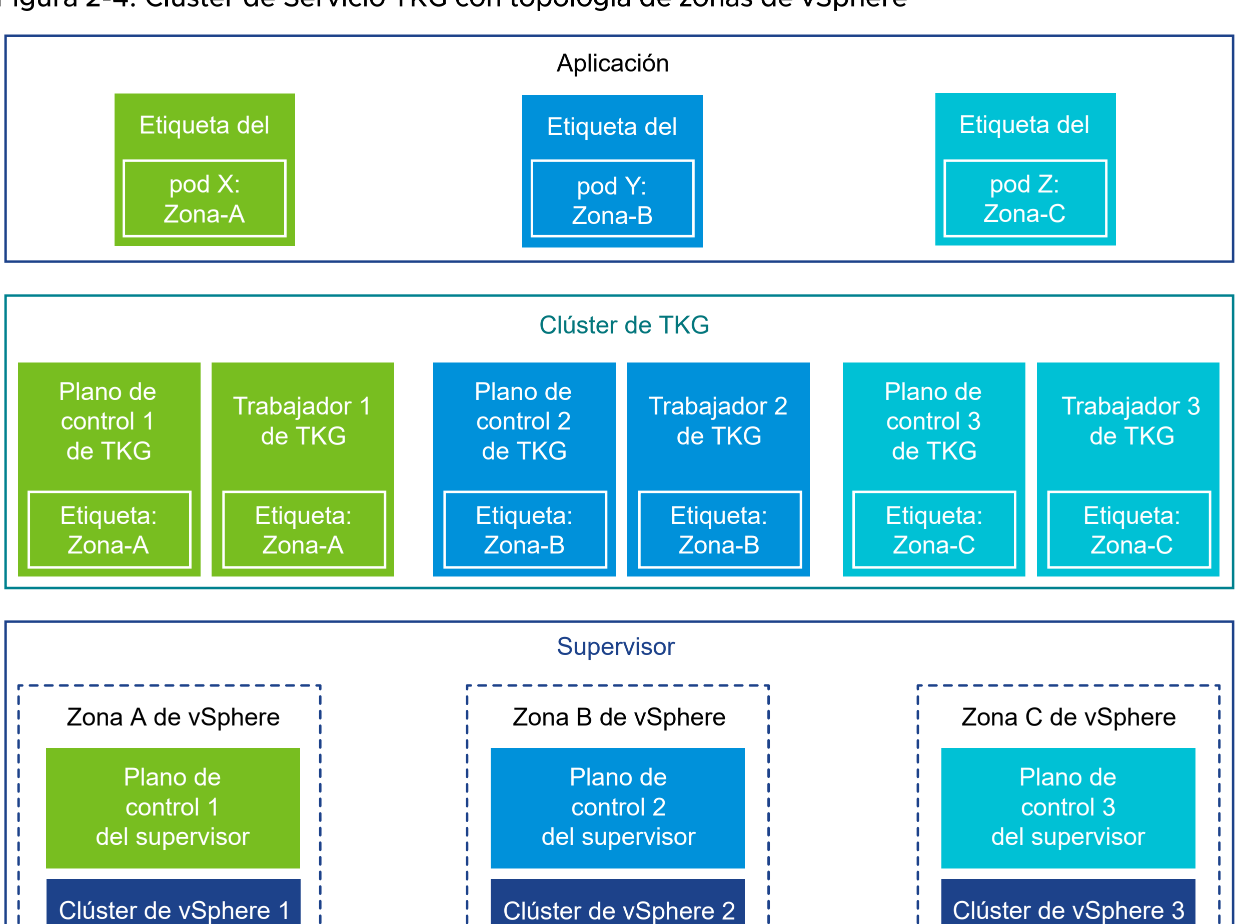
La arquitectura de referencia muestra un clúster de TKG implementado en Zonas de vSphere. Cada zona de vSphere se asigna a un clúster de vSphere, que es una recopilación de hosts ESXi administrados por vCenter y conectados mediante vSphere Distributed Switch, con almacenamiento compartido y calidades de servicio habilitadas.

En una topología de zonas se implementa Supervisor en tres Zonas de vSphere. El sistema localiza un plano de control de Supervisor en cada zona de vSphere, lo que proporciona alta disponibilidad en caso de error.

Cuando se aprovisiona un clúster de TKG en Supervisor, el clúster reconoce las Zonas de vSphere. La topología de zonas admite dominios de errores para las cargas de trabajo de alta disponibilidad. Si es necesario puede ejecutar una carga de trabajo en una zona específica mediante anotaciones.

**Nota** Las zonas de vSphere son nuevas para vSphere 8. Por lo tanto, las zonas de vSphere solo admiten nuevas implementaciones de Supervisor y clústeres de TKG.

Figura 2-4. Clúster de Servicio TKG con topología de zonas de vSphere



# Instalar y actualizar Servicio TKG

3

En esta sección, se proporciona información para instalar y actualizar Servicio TKG.

Lea los siguientes temas a continuación:

- Usar Servicio TKG
- Compruebe el estado del Servicio TKG
- Registrar una nueva versión de Servicio TKG
- Actualizar la versión de Servicio TKG
- Solucionar problemas del Servicio TKG

## Usar Servicio TKG

El servicio Tanzu Kubernetes Grid de VMware (Servicio TKG) le permite implementar clústeres de carga de trabajo de Kubernetes en el vSphere IaaS control plane. El Servicio TKG proporciona versiones independientes y actualizaciones asincrónicas sin interrupciones de la carga de trabajo.

### Introducción al Servicio TKG

A partir de vSphere 8.0 Update 3, Tanzu Kubernetes Grid se instala como un servicio de supervisor. Este cambio de arquitectura desacopla TKG de las versiones de vSphere IaaS control plane y permite actualizar Servicio TKG independientemente de vCenter Server y Supervisor.

Servicio TKG 3.0 está instalado y se ejecuta en nodos del plano de control de Supervisor. El Servicio TKG se proporciona como una colección anidada de paquetes Carvel. Como servicio de supervisor principal, el Servicio TKG se puede actualizar, incluso en entornos restringidos de Internet, pero no se puede desinstalar ni cambiar a una versión anterior. Puede supervisar y administrar los Servicio TKG desde la pestaña **Administración de cargas de trabajo > Servicios**. Consulte Actualizar la versión de Servicio TKG.

Servicio TKG versión 3.1 será la primera versión independiente a la que se actualice. El Registro de nuevas versiones de Servicio TKG con vCenter y la Actualizar la versión de Servicio TKG de Servicio TKG son procesos independientes.

## Instalación de Servicio TKG 3.0

La instalación de Servicio TKG se realiza automáticamente cuando se actualizan los componentes de vSphere IaaS control plane a las versiones requeridas. Consulte las notas de la versión de Servicio TKG para obtener más información.

## Registro de nuevas versiones de Servicio TKG con vCenter

El paquete de Servicio TKG se publica con vCenter Server y se inserta en el registro público VMware. El registro de Servicio TKG se realiza en el nivel de vCenter Server. Tiene dos opciones para hacer el registro nuevas versiones del Servicio TKG: sincrónico y asincrónico.

**Tabla 3-1. Opciones de registro de la versión de Servicio TKG**

| Método de registro | Descripción |
| --- | --- |
| Sincrónico | Espere una actualización a la versión más reciente de vCenter Server para registrar automáticamente una nueva versión del Servicio TKG. Entonces, actualice Supervisor para rellenar el registro integrado con las nuevas versiones. |
| Asincrónico | Descargue del registro público una nueva definición de versión de Servicio TKG. Entonces, regístrela manualmente con vCenter Server. |

El registro sincrónico requiere una actualización del sistema. Al actualizar vCenter Server, se registran automáticamente las nuevas versiones de Servicio TKG con Supervisor. No obstante, para utilizar una versión registrada automáticamente (nueva), debe actualizar Supervisor a la versión que se incluye con la versión de espacios de nombres de vSphere proporcionada por el vCenter Server. Al actualizar Supervisor, el paquete de Carvel para el Servicio TKG está disponible en el registro integrado del Supervisor y listo para la implementación. Una actualización del Supervisor no actualiza automáticamente el Servicio TKG. Debe elegir implementar la versión que desee.

El registro asincrónico no requiere actualizaciones de vCenter Server ni del Supervisor, suponiendo que la versión actual del Supervisor se encuentre dentro de la ventana de soporte. El registro asincrónico tiene el siguiente flujo de trabajo:

1 Descargue el archivo YAML de definición del servicio del sitio de registro público para servicios de supervisor.

2 Registre la nueva versión del Servicio TKG cargando la definición del servicio en vCenter Server.

En la tabla se resumen los detalles de registro del Servicio TKG.

Tabla 3-2. Registro de la versión de Servicio TKG

| Propiedad del servicio TKG | Paquete de vCenter | Registro público |
|---|---|---|
| Registro de nuevas versiones | Registrado automáticamente | Registro manual |
| Eliminación de las versiones registradas recientemente | No permitido | Permitido |
| Ubicación de la imagen | Registro integrado del plano de control del supervisor | Registro público |

## Actualizar la versión de Servicio TKG

Las actualizaciones de la versión del Servicio TKG se realizan en el nivel del Supervisor. Una vez que esté registrado el Servicio TKG, actualice el Servicio TKG implementándolo como un servicio de supervisor en el Supervisor de destino. Consulte Actualizar la versión de Servicio TKG.

Para actualizar Servicio TKG en un entorno restringido de Internet ("aislado"), debe registrar la nueva versión de Servicio TKG de forma sincrónica mediante la actualización de vCenter Server. Al seleccionar la versión que desea instalar, se utiliza el registro local para instalar la nueva versión de Servicio TKG. Consulte Registro de nuevas versiones de Servicio TKG con vCenter.

Cuando se actualiza la versión de Servicio TKG, el sistema realiza comprobaciones previas y notifica dos niveles de gravedad:

- ADVERTENCIA, sin bloqueo
- ERROR, con bloqueo

Una comprobación de la versión de Kubernetes es un ejemplo de comprobación con advertencia sin bloqueo. Una comprobación de la versión de Supervisor es un ejemplo de error con bloqueo. Para obtener más información, consulte la documentación de servicios de Supervisor.

# Compruebe el estado del Servicio TKG

Consulte este tema para comprobar el estado del Servicio TKG.

Complete esta tarea para comprobar que el Servicio TKG esté instalado como un servicio de supervisor principal y para comprobar su estado.

Existen dos formas de comprobar el estado: usar el vSphere Client y usar kubectl. Para comprobar el estado mediante el vSphere Client, inicie sesión en vCenter Server y desplácese hasta **Administración de cargas de trabajo > Servicios**.

Para comprobar el estado mediante kubectl, realice los siguientes pasos.

### Requisitos previos

En esta tarea, se asume que actualizó todos los componentes del sistema e instaló Servicio TKG 3.0. Consulte #unique_67/unique_67_Connect_42_TABLE_CED10728-0714-4D00-BF58-D4BBAA2A4D8E.

### Procedimiento

1 Inicie sesión en Supervisor mediante kubectl.

```
kubectl vsphere login --server=<SUPERVISOR-IP-or-FQDN> --vsphere-username <VCENTER-SSO-
USER>
```

2 Ejecute el siguiente comando.

```
kubectl get packageinstall --namespace vmware-system-supervisor-services
```

Debería ver que el Servicio TKG está instalado.

```
NAMESPACE                           NAME                            PACKAGE NAME
PACKAGE VERSION   DESCRIPTION         AGE
vmware-system-supervisor-services   svc-
tkg.vsphere.vmware.com  tkg.vsphere.vmware.com  0.0.1-b836be7    Reconcile succeeded  17h
```

# Registrar una nueva versión de Servicio TKG

Consulte este tema para registrar de forma manual una nueva versión de Servicio TKG con vCenter Server a fin de actualizar Servicio TKG de forma asincrónica.

Esta tarea solo es necesaria si pretende actualizar la versión de Servicio TKG de forma asincrónica. Consulte Registro de nuevas versiones de Servicio TKG con vCenter.

### Requisitos previos

En esta tarea, se asume que actualizó todos los componentes del sistema e instaló Servicio TKG 3.0. Consulte Instalación de Servicio TKG 3.0.

### Procedimiento

1 En un navegador, vaya al sitio de distribución de servicios de supervisor: https://www.vmware.com/go/supervisor-service.

2 Descargue el archivo `package.yaml` de Servicio TKG del sitio.

3 Mediante vSphere Client, inicie sesión en vCenter Server.

4 Desplácese hasta **Administración de cargas de trabajo > Servicios**.

5 Busque el mosaico de servicio con el nombre **Tanzu Kubernetes Grid Service**.

6 Seleccione **Acciones > Agregar nueva versión**.

7 Haga clic en **Cargar**.

8 Seleccione el archivo `package.yaml` que descargó.

9 Haga clic en **Finalizar**.

#### Resultados

Después de registrar la nueva definición de servicio, es posible que vea más de una versión de Servicio TKG disponible para el mosaico de Servicio TKG. Seleccione la versión de destino cuando actualice Servicio TKG.

#### Pasos siguientes

Actualizar la versión de Servicio TKG.

## Actualizar la versión de Servicio TKG

Consulte este tema para actualizar la versión de Servicio TKG.

Complete los pasos siguientes para actualizar la versión de Servicio TKG.

La actualización de la versión del Servicio TKG se realiza en el nivel del Supervisor. Si vCenter Server aloja varios Supervisores, debe seleccionar el Supervisor de destino.

#### Requisitos previos

En esta tarea, se asume que actualizó todos los componentes del sistema e instaló Servicio TKG 3.0. Consulte Instalación de Servicio TKG 3.0.

En esta tarea se supone que ya ha registrado una nueva versión del Servicio TKG, ya sea Registrar una nueva versión de Servicio TKG o Registro de nuevas versiones de Servicio TKG con vCenter.

#### Procedimiento

1 Mediante vSphere Client, inicie sesión en vCenter Server.

2 Desplácese hasta **Administración de cargas de trabajo > Servicios**.

3 Busque el mosaico de **Tanzu Kubernetes Grid Service**.

4 Seleccione **Acciones > Instalar en supervisores**.

5 Seleccione la versión de destino del Servicio TKG a la que desea actualizar.

6 Seleccione el Supervisor de destino que aloja el Servicio TKG que desea actualizar.

7 Confirme la compatibilidad del servicio y haga clic en **Aceptar**.

8 Confirme que el Servicio TKG esté actualizado.

   El mosaico **Tanzu Kubernetes Grid Service** de la página **Administración de cargas de trabajo > Servicios** reflejará la versión y el estado. También puede comprobar el estado mediante `kubectl get tkr`.

## Solucionar problemas del Servicio TKG

Consulte este tema para solucionar los problemas del Servicio TKG.

## Paquete de soporte de Servicio TKG

El paquete de soporte de Servicio TKG se incluye con el paquete de soporte de Supervisor. Para obtener instrucciones, consulte Recopilar un paquete de soporte para Supervisor.

Dentro del paquete de soporte de Supervisor, los registros de Servicio TKG se encuentran en la carpeta `var/log/tkg-svs`.

servicios de supervisor y administrados por el controlador de servicio principal, los registros de la controladora de servicio principal se encuentran en `/var/log/vmware/wcp/`.

Los registros de la plataforma de aplicaciones se pueden extraer mediante el siguiente comando.

```
kubectl logs vmware-system-appplatform-lifecycle-xxx -n vmware-system-appplatform-operator-
system
```

# Configurar identidad y acceso de los clústeres de Servicio TKG

4

En esta sección se proporciona información para configurar la autenticación y la autorización de usuarios para los clústeres de Servicio TKG en Supervisor.

Lea los siguientes temas a continuación:

- Acerca de la administración de identidades y acceso para clústeres de Servicio TKG
- Instalar las herramientas de CLI para clústeres de Servicio TKG
- Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO
- Conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo
- Conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes
- Crear un grupo y una función dedicados para los operadores de plataforma

## Acerca de la administración de identidades y acceso para clústeres de Servicio TKG

Los ingenieros de desarrollo y operaciones se conectan a vSphere IaaS control plane para aprovisionar y administrar el ciclo de vida de los clústeres de Servicio TKG. Los desarrolladores se conectan a los clústeres de Servicio TKG para implementar paquetes, cargas de trabajo y servicios. Es posible que los administradores necesiten acceso directo a los nodos de clúster de Servicio TKG para solucionar problemas. La plataforma brinda herramientas y métodos de administración de identidades y acceso para respaldar cada caso práctico y cada función.

### El acceso al clúster de Servicio TKG se limita al ámbito de espacio de nombres de vSphere

Los clústeres de Servicio TKG se aprovisionan en espacio de nombres de vSphere. Al configurar un espacio de nombres de vSphere, se establecen permisos de desarrollo y operaciones, incluidos el origen de identidad, los usuarios y los grupos, y las funciones. La plataforma propaga estos permisos a cada clúster de Servicio TKG aprovisionado en ese espacio de nombres de vSphere. La plataforma admite dos métodos de autenticación: vCenter Single Sign-On y un proveedor de identidad externo compatible con OIDC.

## Autenticación mediante vCenter Single Sign-On y Kubectl

De forma predeterminada, vCenter Single Sign-On se utiliza para autenticarse con el entorno, incluso los clústeres de Supervisor y Servicio TKG. vCenter Single Sign-On proporciona autenticación para la infraestructura de vSphere y se puede integrar con sistemas AD/LDAP. Para obtener más información, consulte Autenticación de vSphere con vCenter Single Sign-On.

Para autenticarse con vCenter Single Sign-On, utilice complemento de vSphere para kubectl. Una vez que se haya autenticado, utilice kubectl para aprovisionar y administrar de forma declarativa el ciclo de vida de los clústeres de servicio TKG e interactuar con Supervisor.

El complemento complemento de vSphere para kubectl depende de kubectl. Cuando se autentica con el comando `kubectl vsphere login`, el complemento realiza una solicitud POST con autenticación básica en un endpoint /wcp/login en Supervisor. vCenter Server emite un token web JSON (JWT) en el que Supervisor confía.

Para conectarse mediante vCenter Single Sign-On, consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

## Autenticación mediante un proveedor de identidad externo y la CLI de Tanzu

Es posible configurar Supervisor con un proveedor de identidad externo que admita el protocolo OpenID Connect. Una vez configurado, Supervisor funciona como un cliente de OAuth 2.0 y utiliza el servicio de autenticación Pinniped para proporcionar conectividad al cliente mediante la CLI de Tanzu. CLI de Tanzu admite el aprovisionamiento y la administración del ciclo de vida de los clústeres de Servicio TKG. Cada instancia de Supervisor puede admitir un solo proveedor de identidad externo.

Una vez que el complemento de autenticación y el emisor de OIDC están configurados correctamente para que funcione la CLI de autenticación de Pinniped, al iniciar sesión en Supervisor mediante `tanzu login --endpoint`, el sistema busca algunos configmaps conocidos para compilar el archivo de configuración `pinniped config`.

Para conectarse mediante un proveedor de OIDC externo, consulte Conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo.

## Autenticación mediante un enfoque híbrido: vCenter SSO con la CLI de Tanzu

Si utiliza vCenter Single Sign-On como proveedor de identidad y desea utilizar CLI de Tanzu, puede adoptar un enfoque híbrido e iniciar sesión en Supervisor mediante ambas herramientas. Este enfoque puede resultar útil para instalar paquetes estándar. Consulte Conectarse a Supervisor mediante la CLI de Tanzu y la autenticación de vCenter SSO.

## Usuarios y grupos para desarrollo y operaciones

Los permisos que establece al configurar un espacio de nombres de vSphere son para que los usuarios de desarrollo y operaciones administren el ciclo de vida de los clústeres de Servicio TKG. El usuario o el grupo de desarrollo y operaciones al que asigne permisos deben existir en el origen de identidad. Los usuarios de desarrollo y operaciones se autentican con sus credenciales de proveedor de identidad.

Los usuarios de desarrollo y operaciones son responsables de conceder acceso al clúster a los usuarios de procesos posteriores, como los desarrolladores que desean implementar cargas de trabajo en clústeres aprovisionados. Estos usuarios se autenticarían con un proveedor de identidad o con una función de clúster y un enlace en Kubernetes. Esto se describe más detalladamente en la siguiente sección.

**Nota** Los permisos de espacio de nombres de vSphere son exclusivos para los usuarios de desarrollo y operaciones que necesitan crear y administrar clústeres de Servicio TKG. Los usuarios de estos clústeres, como los desarrolladores, utilizarán mecanismos de autenticación de Kubernetes.

## Enlaces y permisos de función

Existen dos tipos de sistemas de control de acceso basado en funciones (Role Based Access Control, RBAC) para los clústeres de TKGS: permisos de espacio de nombres de vSphere y autorización de RBAC de Kubernetes. Como administrador de vSphere, debe asignar permisos de espacio de nombres de vSphere para que los usuarios puedan crear y utilizar clústeres de servicio TKG. Los operadores de clúster utilizan el RBAC de Kubernetes para conceder acceso al clúster y asignar permisos de función a los desarrolladores. Consulte Conceder a los desarrolladores acceso de vCenter SSO a los clústeres de Servicio TKG.

espacios de nombres de vSphere admite tres funciones: **Puede editar**, **Puede ver** y **Propietario**. La asignación y el ámbito de los permisos de función dependen de la instancia de espacio de nombres de vSphere que aloja un clúster de Servicio TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

Un usuario o grupo con el permiso de función **Puede editar** concedido en un espacio de nombres de vSphere puede crear, leer, actualizar y eliminar clústeres de Servicio TKG en ese espacio de nombres de vSphere. Además, cuando se asigna un usuario o un grupo a la función **Puede editar**, el sistema crea un objeto RoleBinding en cada clúster de esa instancia de espacio de nombres de vSphere y enlaza el privilegio al objeto ClusterRole de Kubernetes denominado

`cluster-admin`. La función `cluster-admin` permite a los usuarios aprovisionar y operar clústeres de servicio TKG en la instancia de espacio de nombres de vSphere de destino. Puede ver esta asignación mediante el comando `kubectl get rolebinding` desde el espacio de nombres de vSphere de destino.

```
kubectl get rolebinding -n tkgs-cluster-namespace
NAME                                                               ROLE
AGE
wcp:tkg-cluster-namespace:group:vsphere.local:administrators   ClusterRole/edit
33d
wcp:tkg-cluster-namespace:user:vsphere.local:administrator     ClusterRole/edit
33d
```

Un usuario o un grupo que tiene asignado el permiso de función **Puede ver** en una instancia de espacio de nombres de vSphere tiene acceso de solo lectura a los clústeres de Servicio TKG aprovisionados en esa instancia de espacio de nombres de vSphere. Sin embargo, a diferencia del privilegio **Puede editar**, para la función **Puede ver**, no se creó ningún objeto RoleBinding de Kubernetes en los clústeres de TKGS de esa instancia de espacio de nombres de vSphere. Esto se debe a que en Kubernetes no hay ninguna función equivalente de solo lectura a la que pueda estar enlazado un usuario o grupo que tenga concedido el permiso **Puede ver**. Para los usuarios que no sean `cluster-admin`, utilice RBAC de Kubernetes para conceder acceso. Consulte Conceder a los desarrolladores acceso de vCenter SSO a los clústeres de Servicio TKG.

Un usuario o grupo con el permiso **Propietario** concedido en un espacio de nombres de vSphere puede administrar clústeres de Servicio TKG en ese espacio de nombres de vSphere, y puede crear y eliminar espacios de nombres de vSphere adicionales mediante kubectl. Cuando se asigna un usuario o grupo a la función de propietario, el sistema crea un objeto ClusterRoleBinding y lo asigna a un objeto ClusterRole que permite al usuario o grupo crear y eliminar espacios de nombres de vSphere mediante kubectl. No es posible ver esta asignación con el mismo enfoque. Debe utilizar SSH en un nodo de Supervisor para ver esta asignación.

**Nota** La función de propietario es compatible con los usuarios que provienen del origen de identidad de vCenter Single Sign-On. No se puede usar la función de propietario con un usuario o grupo que provenga de un proveedor de identidad externo.

## Permisos de vSphere

La tabla muestra los diferentes permisos de vSphere necesarios para los distintos roles de vSphere IaaS control plane. Si es necesario, puede crear un grupo de SSO de vSphere personalizado y una función para la administración de cargas de trabajo. Consulte Crear un grupo y una función dedicados para los operadores de plataforma.

Tabla 4-1. Permisos de vSphere para roles de vSphere with Tanzu

| Persona | Función de vSphere | Grupo de vSphere SSO | Espacio de nombres de vSphere |
| --- | --- | --- | --- |
| Administrador de VI/Cloud | Administrador | Administradores | Usuario de SSO o usuario de AD |
| Operador de plataforma/ desarrollo y operaciones | Función personalizada o no administrativa | ServiceProviderUsers | Usuario de SSO o usuario de AD |
| Desarrollador | Solo lectura o Ninguno | Ninguno | Usuario de SSO o usuario de AD |

## Conectividad del administrador del sistema

Los administradores pueden conectarse a los nodos de clúster de Servicio TKG como el usuario `kubernetes-admin`. Este método puede resultar apropiado si la autenticación de vCenter Single Sign-On no está disponible. Con el fin de solucionar problemas, los administradores de sistemas pueden conectarse a Servicio TKG como `vmware-system-user` mediante SSH y una clave privada. Consulte Conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes.

# Instalar las herramientas de CLI para clústeres de Servicio TKG

vSphere IaaS control plane proporciona varias herramientas de CLI para conectarse a clústeres de Servicio TKG y administrar su ciclo de vida.

## Instalar el Herramientas de la CLI de Kubernetes para vSphere

Los usuarios de vCenter Single Sign-On pueden utilizar Herramientas de la CLI de Kubernetes para vSphere para conectarse a los clústeres de Servicio TKG e interactuar con ellos.

### Acerca de las Herramientas de la CLI de Kubernetes para vSphere

El paquete de descarga de Herramientas de la CLI de Kubernetes para vSphere (`vsphere-plugin.zip`) incluye dos ejecutables: kubectl y el complemento de vSphere para kubectl.

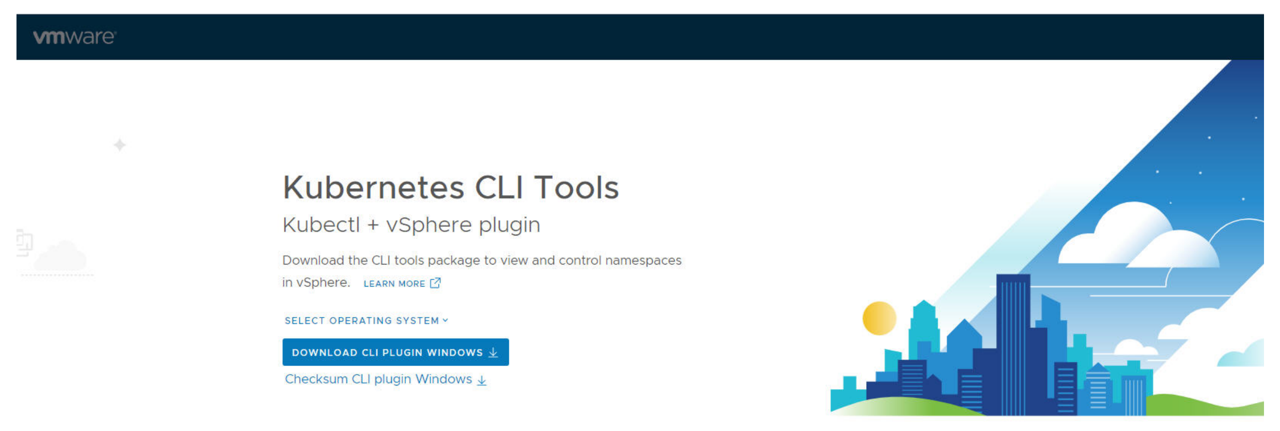
La CLI de kubectl tiene una arquitectura acoplable. El complemento de vSphere para kubectl extiende los comandos disponibles a kubectl de modo que sea posible conectarse al Supervisor y a los clústeres de TKG mediante las vCenter Single Sign-On.

**Nota** vSphere IaaS control plane proporciona archivos binarios para los procesadores x86/64.

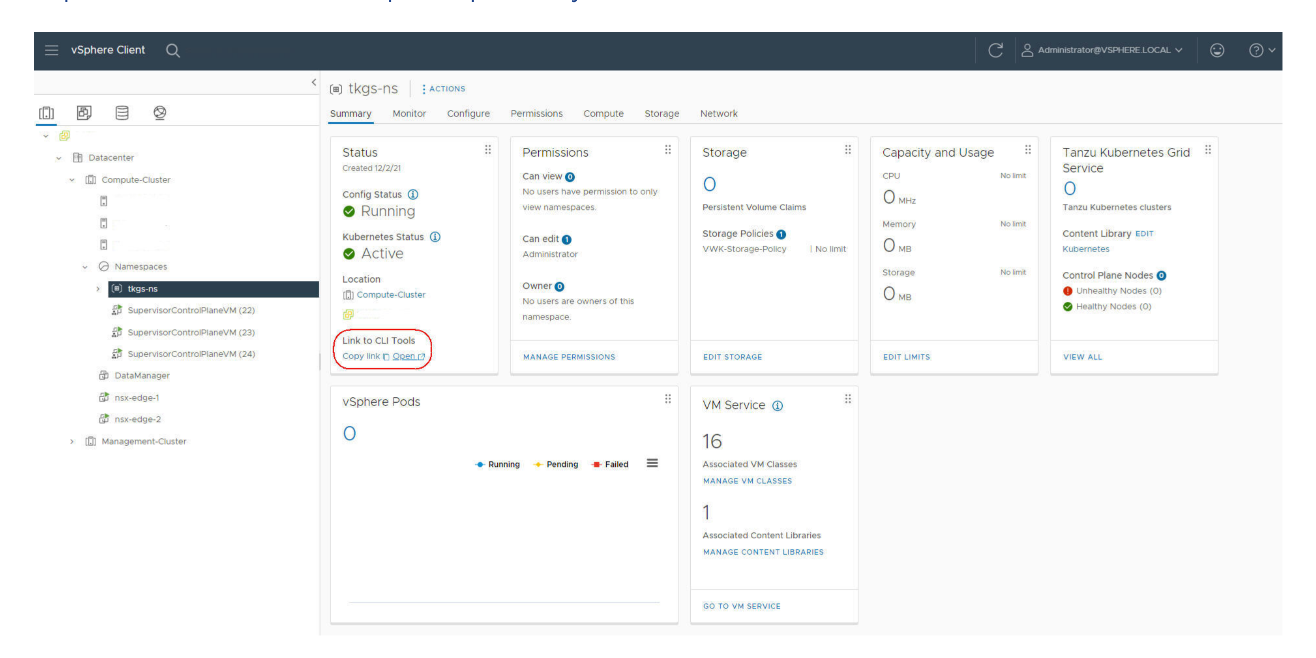
### Requisito previo: se crea espacio de nombres de vSphere

Las Herramientas de la CLI de Kubernetes para vSphere están disponibles para su descarga en la página de desarrollo y operaciones de vSphere IaaS control plane.

Figura 4-1. Página de desarrollo y operaciones de vSphere




Puede acceder a la página de desarrollo y operaciones de vSphere IaaS control plane desde el panel de configuración de espacio de nombres de vSphere. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.




## Instalar las Herramientas de la CLI de Kubernetes para vSphere mediante vSphere Client

Para instalar las Herramientas de la CLI de Kubernetes para vSphere desde la página de desarrollo y operaciones de vCenter, siga estos pasos.

1 Obtenga de su administrador de vSphere el vínculo de la página de descargas Herramientas de la CLI de Kubernetes. De forma alternativa, si tiene acceso a vCenter Server, puede obtener el vínculo de la siguiente manera.

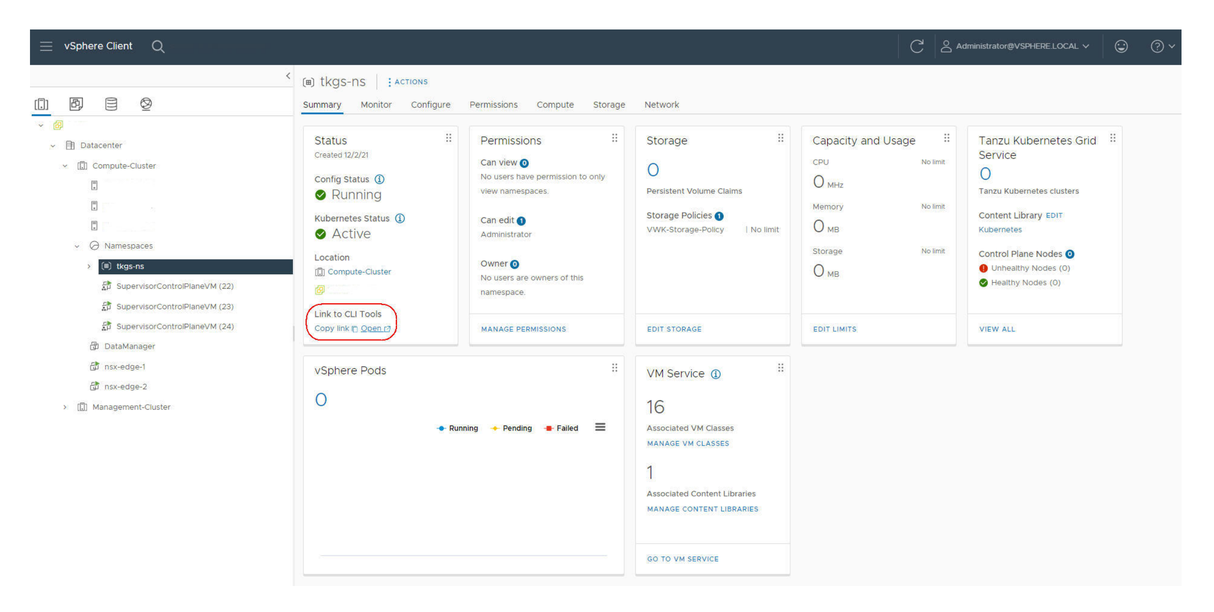
   a Inicie sesión en vCenter Server mediante vSphere Client.

   b Desplácese hasta **Administración de cargas de trabajo > Espacios de nombres**.

   c Seleccione el espacio de nombres de vSphere en el que está trabajando.

d Seleccione la pestaña **Resumen** y encuentre el área **Estado** en esta página. (Consulte la imagen).

e Seleccione **Abrir** debajo del encabezado **Vínculo a herramientas de CLI** para abrir la página de descargas. O bien, puede **Copiar** el vínculo.




2 Seleccione el sistema operativo.

---

**Nota** Consulte los pasos de instalación específicos del sistema operativo al final de este tema según sea necesario.

---

3 Descargue el archivo `vsphere-plugin.zip`.

4 Extraiga el contenido del archivo ZIP en un directorio de trabajo.

5 Agregue la ubicación de los dos archivos ejecutables a la variable PATH del sistema.

6 Para comprobar la instalación de la CLI de kubectl, inicie una sesión de Shell, de terminal o de línea de comandos y ejecute el comando `kubectl`.

Verá el mensaje de aviso de kubectl y la lista de opciones de línea de comandos para la CLI.

7 Para comprobar la instalación de complemento de vSphere para kubectl, ejecute el comando `kubectl vsphere`.

Verá el mensaje de aviso de complemento de vSphere para kubectl y la lista de opciones de línea de comandos para el complemento.

## Instalar las Herramientas de la CLI de Kubernetes para vSphere desde la línea de comandos

Si utiliza Linux o MacOS, puede ejecutar el siguiente comando para descargar las Herramientas de la CLI de Kubernetes para vSphere.

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo > Supervisor**.

3 Obtenga la dirección IP del equilibrador de carga para el plano de control de Supervisor.

4 Cree una variable para ella.

5 Ejecute el siguiente comando para instalar las herramientas. (Es posible que deba ajustarlo para su entorno).

Linux:

```
curl -LOk https://${SC_IP}/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

MacOS:

```
curl -LOk https://${SC_IP}/wcp/plugin/macos-darwin64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

6 Ejecute los comandos `kubectl` y `kubectl vsphere`, y compruebe la instalación.

## Instalar las Herramientas de la CLI de Kubernetes para vSphere en Linux

Instale las Herramientas de la CLI de Kubernetes para vSphere en un host Linux.

1 Descargue el archivo `vsphere-plugin.zip` para Linux.

2 Extraiga el contenido del archivo, que son dos archivos ejecutables: kubectl y `kubectl-vsphere`.

3 Coloque los dos archivos ejecutables en la ruta de búsqueda ejecutable del sistema operativo, como `/usr/local/bin`.

4 Ejecute el comando `kubectl vsphere` para comprobar la instalación.

5 Ejecute el comando `kubectl vsphere login --server=Supervisor-IP` para iniciar sesión en Supervisor.

6 Ejecute el comando `kubectl config get-contexts` para ver una lista de los espacios de nombres de vSphere a los que tiene acceso.

7 Ejecute el comando `kubectl config use-context CONTEXT` para elegir el contexto predeterminado.

## Instalar las Herramientas de la CLI de Kubernetes para vSphere en MacOS

Instale las Herramientas de la CLI de Kubernetes para vSphere en un host MacOS.

1 Descargue el archivo `vsphere-plugin.zip` para MacOS.

2 Extraiga el contenido del archivo, que son dos archivos ejecutables: kubectl y `kubectl-vsphere`.

3 Coloque los dos archivos ejecutables en la ruta de búsqueda ejecutable del sistema operativo, como `/usr/local/bin`.

4 Ejecute el comando `kubectl vsphere` para comprobar la instalación.

5 Ejecute el comando `kubectl vsphere login --server=Supervisor-IP` para iniciar sesión en Supervisor.

6 Ejecute el comando `kubectl config get-contexts` para ver una lista de los espacios de nombres de vSphere a los que tiene acceso.

7 Ejecute el comando `kubectl config use-context CONTEXT` para elegir el contexto predeterminado.

## Instalar las Herramientas de la CLI de Kubernetes para vSphere en Windows

Instale las Herramientas de la CLI de Kubernetes para vSphere en un host Windows.

1 Descargue el archivo `vsphere-plugin.zip` para MacOS.

2 Extraiga el contenido del archivo, que son dos archivos ejecutables: `kubectl.exe` y `kubectl-vsphere.exe`.

3 Coloque los dos archivos ejecutables en la ruta del sistema.

4 Ejecute el comando `kubectl vsphere` para comprobar la instalación.

5 Ejecute el comando `kubectl vsphere login --server=Supervisor-IP` para iniciar sesión en Supervisor.

6 Ejecute el comando `kubectl config get-contexts` para ver una lista de los espacios de nombres de vSphere a los que tiene acceso.

7 Ejecute el comando `kubectl config use-context CONTEXT` para elegir el contexto predeterminado.

# Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG

Puede utilizar CLI de Tanzu para aprovisionar y administrar el ciclo de vida de los clústeres de Servicio TKG.

Instale CLI de Tanzu para trabajar con clústeres de Servicio TKG.

### Procedimiento

1 Consulte en la documentación de producto de la CLI de Tanzu las instrucciones de instalación.

2 Compruebe la matriz de compatibilidad para obtener la versión adecuada.

**Nota** Para Servicio TKG 3.0, no utilice una versión de la CLI de Tanzu posterior a 1.1.0. Por ejemplo, no utilice la CLI de Tanzu 1.2.0.

**Nota** A partir del servicio TKG 3.1.0, descargue el complemento de CLI de autenticación de Pinniped con la misma versión de Servicio TKG y descargue el complemento de CLI imgpkg con la versión más reciente.

3 Para instalar la CLI de Tanzu, siga las instrucciones de la versión de destino.

Por ejemplo: Instalar y usar la CLI de VMware Tanzu v1.1.x: https://docs.vmware.com/es/VMware-Tanzu-CLI/1.1/tanzu-cli/index.html

## Instalar el complemento auxiliar de credenciales de vSphere Docker

Utilice la CLI del complemento auxiliar de credenciales de vSphere Docker para insertar imágenes de contenedor de forma segura en el registro de Harbor integrado y exportarlas desde él.

Utilice el complemento auxiliar de credenciales de vSphere Docker para conectar de forma segura su cliente de Docker a un registro de contenedor.

**Nota** complemento auxiliar de credenciales de vSphere Docker está pensado para utilizarse con el registro de Harbor integrado, que está obsoleto. Si utiliza el servicio de supervisor de Harbor, consulte el siguiente tema para conectarse a él mediante Docker: Configurar un cliente de Docker con el certificado de registro de Harbor.

#### Requisitos previos

La página de descargas Herramientas de la CLI de Kubernetes incluye un vínculo para descargar el complemento auxiliar de credenciales de vSphere Docker.

- Configure un cliente de Docker.
- Obtenga acceso al registro de Harbor integrado habilitado en Supervisor.
- Obtenga de su administrador de vSphere el vínculo de la página de descargas de Herramientas de la CLI de Kubernetes para vSphere.
- De forma alternativa, si tiene acceso a la instancia de vCenter Server, obtenga el vínculo de la siguiente manera:
  - Inicie sesión en vCenter Server mediante vSphere Client.
  - Desplácese hasta **Administración de cargas de trabajo > Espacios de nombres** y seleccione el espacio de nombres de vSphere de destino.
  - Seleccione la pestaña **Resumen** y localice el mosaico de **Estado**.
  - Seleccione **Abrir** debajo del encabezado **Vínculo a herramientas de CLI** para abrir la página de descargas. O bien, puede **Copiar** el vínculo.

Procedimiento

1 Con un navegador, vaya a la URL de descarga de **Herramientas de la CLI de Kubernetes** correspondiente a su entorno.

2 Desplácese hacia abajo hasta la sección complemento auxiliar de credenciales de vSphere Docker.

3 Seleccione el sistema operativo.

4 Descargue el archivo `vsphere-docker-credential-helper.zip`.

5 Extraiga el contenido del archivo ZIP en un directorio de trabajo.

   El archivo ejecutable binario `docker-credential-vsphere` está disponible.

6 Copie el archivo binario `docker-credential-vsphere` en el host del cliente de Docker.

7 Agregue la ubicación del archivo binario a la variable PATH del sistema.

   Por ejemplo, en Linux:

```
mv docker-credential-vsphere /usr/local/bin/docker-credential-vsphere
```

8 Compruebe la instalación del complemento auxiliar de credenciales de vSphere Docker. Para ello, ejecute el comando `docker-credential-vsphere` en un shell o una sesión de terminal.

   Verá el mensaje de aviso y la lista de opciones de línea de comandos para la CLI.

```
vSphere login manager is responsible for vSphere authentication.
It allows vSphere users to securely login and logout to access Harbor images.

Usage:
  docker-credential-vsphere [command]

Available Commands:
  help        Help about any command
  login       Login into specific harbor server and get authentication
  logout      Logout from Harbor server and erase user token

Flags:
  -h, --help   help for docker-credential-vsphere

Use "docker-credential-vsphere [command] --help" for more information about a command.
```

9 Inicie sesión en un registro.

   En primer lugar, compruebe el uso:

```
docker-credential-vsphere login -help
Usage:
  docker-credential-vsphere login [harbor-registry] [flags]

Flags:
```

```
  -h, --help               help for login
  -s, --service string     credential store service
      --tlscacert string   location to CA certificate (default "/etc/docker/certs.d/*.crt")
  -u, --user string        vSphere username and password
```

**Nota** complemento auxiliar de credenciales de vSphere Docker espera que la cadena del usuario tenga todos los caracteres en minúsculas. Si no utiliza todos los caracteres en minúsculas, puede que el inicio de sesión funcione, pero que no lo hagan los comandos de Docker posteriores. Asegúrese de utilizar todos los caracteres en minúsculas para el nombre de usuario.

A continuación, inicie sesión con el siguiente comando:

```
docker-credential-vsphere login <container-registry-IP>
```

Se obtiene el token de autenticación y se guarda. En ese momento se inicia la sesión.

```
docker-credential-vsphere login 10.179.145.77
Username: administrator@vsphere.local
Password: INFO[0017] Fetched username and password
INFO[0017] Fetched auth token
INFO[0017] Saved auth token
```

10 Cierre la sesión del registro de Harbor.

```
docker-credential-vsphere logout 10.179.145.77
```

# Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO

Puede conectarse a clústeres de Servicio TKG mediante vCenter Single Sign-On y el complemento de vSphere para kubectl por sí mismo o con la CLI de Tanzu.

## Configurar el inicio de sesión seguro para la autenticación de vCenter Single Sign-On

Para iniciar sesión de forma segura en Supervisor y en clústeres de Servicio TKG, configure complemento de vSphere para kubectl con el certificado TLS adecuado y asegúrese de ejecutar la edición más reciente del complemento.

### Certificado de CA de Supervisor

vSphere IaaS control plane admite vCenter Single Sign-On para el acceso a los clústeres mediante el comando de complemento de vSphere para kubectl `kubectl vsphere login …`.

El complemento complemento de vSphere para kubectl establece de forma predeterminada el inicio de sesión seguro y requiere un certificado de confianza, el certificado firmado por la entidad de certificación raíz vCenter Server. A pesar de que el complemento es compatible con la marca de `--insecure-skip-tls-verify`, por motivos de seguridad, esto no se recomienda.

Para iniciar sesión de forma segura en los clústeres de Supervisor y Servicio TKG mediante el complemento complemento de vSphere para kubectl, tiene dos opciones:

| Opción | Instrucciones |
| --- | --- |
| Descargue e instale el certificado de la entidad de certificación de vCenter Server raíz en cada máquina cliente. | Para Linux, consulte la siguiente sección: Descargar los certificados de CA raíz de confianza para vCenter e instalarlos en un cliente Ubuntu<br>Para Windows y Mac, consulte el artículo de la base de conocimientos de VMware Cómo descargar e instalar certificados raíz de vCenter Server. |
| Reemplace el certificado VIP utilizado para Supervisor por un certificado firmado por una entidad de certificación de confianza de cada máquina cliente. | Consulte *Instalar y configurar el plano de control de IaaS de vSphere*. |

**Nota** Para obtener más información sobre la autenticación de vSphere, incluidos vCenter Single Sign-On, la administración y rotación de certificados de vCenter Server y la solución de problemas, consulte la documentación de *autenticación de vSphere*.

## Certificado de CA del clúster de TKG

Para conectarse de forma segura con el servidor de API del clúster de TKG mediante la CLI `kubectl`, debe descargar el certificado de CA del clúster de TKG.

Si utiliza la edición más reciente del complemento de vSphere para kubectl, la primera vez que inicie sesión en el clúster de TKG, el complemento registrará el certificado de la entidad de certificación del clúster de TKG en el archivo kubconfig. Este certificado también se almacena en el secreto de Kubernetes denominado `*TANZU-KUBERNETES-CLUSTER-NAME*-ca`. El complemento utiliza el certificado para rellenar la información de CA en el almacén de datos de la entidad de certificación del clúster correspondiente.

Si ha actualizado Supervisor, asegúrese de hacerlo a la versión más reciente del complemento.

## Descargar los certificados de CA raíz de confianza para vCenter e instalarlos en un cliente Ubuntu

Siga este procedimiento para descargar los certificados de CA raíz de confianza para vCenter Server e instalarlos en un cliente Ubuntu a fin de poder iniciar sesión de forma segura en los clústeres de Supervisor y de Servicio TKG mediante complemento de vSphere para kubectl.

1 Instale complemento de vSphere para kubectl. Consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

2 Descargue los certificados de CA raíz de confianza para vCenter Server donde se habilitó **Administración de cargas de trabajo**.

```
wget https://VC-IP-or_FQDN/certs/download.zip --no-check-certificate
```

3 Extraiga el contenido del archivo `download.zip` en el directorio actual.

```
unzip download.zip -d .
```

4 Cambie la ruta de acceso al directorio de Linux.

```
cd /certs/lin
```

5 Enumere (`ls`) los certificados de CA en el directorio `/certs/lin`.

Debería ver dos certificados en formato PEM: `*.0` y `*.r1`. Un certificado con formato PEM se puede leer en formato base64 y comienza con `----BEGIN CERTIFICATE----`.

6 Anexe la extensión `*.crt` a los archivos de certificado. Por ejemplo:

```
cp dbad4059.0 dbad4059.0.crt
```

```
cp dbad4059.r1 dbad4059.r1.crt
```

7 Copie los archivos en el directorio de certificados OpenSSL en `/etc/ssl/certs`.

```
sudo cp dbad4059.0.crt /etc/ssl/certs
```

```
sudo cp dbad4059.r1.crt /etc/ssl/certs
```

8 Inicie sesión de forma segura en Supervisor.

```
kubectl vsphere login --server=IP-or-FQDN --vsphere-username USERNAME
```

9 Inicie sesión de forma segura en el clúster de Servicio TKG.

```
kubectl vsphere login --server=IP-or-FQDN --vsphere-username USERNAME --tanzu-kubernetes-
cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace VSPHERE-NS
```

## Configurar los permisos del espacio de nombres de vSphere para usuarios y grupos de vCenter Single Sign-On

Establezca permisos en el espacio de nombres de vSphere para que los usuarios y los grupos de vCenter Single Sign-On puedan acceder a los clústeres de TKG 2 aprovisionados en estos.

Una vez que haya creado un espacio de nombres de vSphere, agregue usuarios y aplique funciones para configurarlo para los clústeres de TKG 2. Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

#### Requisitos previos

Los permisos de usuarios, grupos y funciones se establecen en el nivel del espacio de nombres de vSphere. Para acceder a Supervisor y los clústeres de TKG 2, primero debe crear un espacio de nombres de vSphere. Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

#### Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo > Espacios de nombres**.

3 Seleccione el espacio de nombres de vSphere que creó.

4 Seleccione **Permisos > Agregar permisos**.

5 **Origen de identidad**: seleccione **vsphere.local** para los usuarios y grupos de vCenter SSO.

   **Nota** Si utiliza un proveedor de identidad externo, consulte Conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo.

6 **Búsqueda de usuarios/grupos**: seleccione el usuario o el grupo de vCenter SSO que se han configurado para las operaciones del clúster de TKG o los desarrolladores de TKG.

7 **Función**: asigne una función al usuario o al grupo mediante la selección de la función adecuada: **Puede ver**, **Puede editar** o **Propietario**.

| Opción | Descripción |
| --- | --- |
| **Puede ver** | Puede leer los objetos del clúster de TKG en el espacio de nombres de vSphere. No hay permisos asignados a las funciones de Kubernetes. Consulte Enlaces y permisos de función. |
| **Puede editar** | Puede crear, leer, actualizar y eliminar objetos de clúster de TKG en el espacio de nombres de vSphere. Puede utilizar clústeres de TKG aprovisionados en vSphere Namespaces como Kubernetes `cluster-admin`. Consulte Enlaces y permisos de función. |
| **Propietario** | Los mismos permisos que Puede editar, con permiso adicional para crear y administrar vSphere Namespaces mediante kubectl. Solo disponible con vCenter SSO. Consulte Enlaces y permisos de función. |

8 Complete la configuración del espacio de nombres de vSphere. Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

## Conectarse a Supervisor como usuario de vCenter Single Sign-On con kubectl

Conéctese al Supervisor mediante el complemento de vSphere para kubectl y autentíquese con las credenciales de vCenter Single Sign-On.

Después de iniciar sesión en Supervisor, el complemento de vSphere para kubectl genera el contexto del clúster. En Kubernetes, un contexto de configuración incluye un clúster, un espacio de nombres y un usuario. Puede ver el contexto del clúster en el archivo `.kube/config`. Generalmente, este archivo se denomina `kubeconfig`.

**Nota** Si ya tiene un archivo `kubeconfig`, este se anexa a cada contexto de clúster. El complemento de vSphere para kubectl respeta la variable de entorno KUBECONFIG que kubectl utiliza. Aunque no es obligatorio, puede que resulte útil definir esta variable antes de ejecutar `kubectl vsphere login ...` para que la información se escriba en un archivo nuevo (en lugar de agregarse al archivo `kubeconfig` actual).

Requisitos previos

- Instalar el Herramientas de la CLI de Kubernetes para vSphere.
- Obtenga las credenciales de vCenter Single Sign-On.
- Obtenga de su administrador de vSphere la dirección IP del plano de control del Supervisor.
- Obtenga el nombre del espacio de nombres de vSphere de su administrador de vSphere.
- Compruebe que tiene permisos **Editar** en el espacio de nombres de vSphere. Consulte Configurar los permisos del espacio de nombres de vSphere para usuarios y grupos de vCenter Single Sign-On.
- Para comprobar que el certificado ofrecido por el plano de control de Kubernetes sea de confianza en el sistema, instale la CA de firma como raíz de confianza o agregue el certificado directamente como raíz de confianza. Consulte Configurar el inicio de sesión seguro para la autenticación de vCenter Single Sign-On.

Procedimiento

1 Para ver la sintaxis y las opciones de los comandos para iniciar sesión, ejecute el siguiente comando.

```
kubectl vsphere login --help
```

2 Para conectarse a Supervisor, ejecute el siguiente comando.

```
kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username
<VCENTER-SSO-USER>
```

También puede iniciar sesión con un FQDN:

```
kubectl vsphere login --server <KUBERNETES-CONTROL-PLANE-FQDN --vsphere-username <VCENTER-
SSO-USER>
```

Por ejemplo:

```
kubectl vsphere login --server=10.92.42.13 --vsphere-username administrator@example.com
```

```
kubectl vsphere login --server wonderland.acme.com --vsphere-username
administrator@example.com
```

Esta acción crea un archivo de configuración con el token web de JSON (JSON Web Token, JWT) para autenticarse en la API de Kubernetes.

3 Para autenticarse, introduzca la contraseña del usuario.

Después de conectarse a Supervisor, se le mostrarán los contextos de configuración a los que puede acceder. Por ejemplo:

```

You have access to the following contexts:
tanzu-ns-1
tkg-cluster-1
tkg-cluster-2
```

4 Para ver los detalles de los contextos de configuración a los que puede acceder, ejecute el siguiente comando de `kubectl`:

```
kubectl config get-contexts
```

La CLI muestra los detalles de cada contexto disponible.

5 Para cambiar de contexto, utilice el siguiente comando:

```
kubectl config use-context <example-context-name>
```

## Conectarse a un clúster de Servicio TKG como usuario de vCenter Single Sign-On con Kubectl

Puede conectarse a un clúster de TKG mediante el complemento de vSphere para kubectl y autenticarse con las credenciales de vCenter Single Sign-On.

Después de iniciar sesión en el clúster de Tanzu Kubernetes, el complemento de vSphere para kubectl genera el contexto del clúster. En Kubernetes, un contexto de configuración incluye un clúster, un espacio de nombres y un usuario. Puede ver el contexto del clúster en el archivo `.kube/config`. Generalmente, este archivo se denomina `kubeconfig`.

---

**Nota** Si ya tiene un archivo `kubeconfig`, este se anexa a cada contexto de clúster. El complemento de vSphere para kubectl respeta la variable de entorno KUBECONFIG que kubectl utiliza. Aunque no es obligatorio, puede que resulte útil definir esta variable antes de ejecutar `kubectl vsphere login ...` para que la información se escriba en un archivo nuevo (en lugar de agregarse al archivo `kubeconfig` actual).

---

### Requisitos previos

Obtenga la siguiente información del administrador de vSphere:

- Obtenga las credenciales de vCenter Single Sign-On.
- Obtenga la dirección IP del plano de control de Supervisor.
- Obtenga el nombre de la instancia de espacio de nombres de vSphere.
- Instalar el Herramientas de la CLI de Kubernetes para vSphere.

### Procedimiento

1 Para ver la sintaxis y las opciones de los comandos para iniciar sesión, ejecute el siguiente comando.

```
kubectl vsphere login --help
```

2 Para conectarse al clúster de Tanzu Kubernetes, ejecute el siguiente comando.

```
kubectl vsphere login --server=SUPERVISOR-CLUSTER-CONTROL-PLANE-IP-OR-FQDN
--tanzu-kubernetes-cluster-name TKG-CLUSTER-NAME
--tanzu-kubernetes-cluster-namespace VSPHERE-NAMESPACE
--vsphere-username VCENTER-SSO-USER-NAME
```

Por ejemplo:

```
kubectl vsphere login --server=10.92.42.137
--tanzu-kubernetes-cluster-name tkg-cluster-01
--tanzu-kubernetes-cluster-namespace tkg-cluster-ns
--vsphere-username operator@example.com
```

También, si se habilitó Supervisor con un nombre de dominio completo (FQDN):

```
kubectl vsphere login --server=wonderland.acme.com
--tanzu-kubernetes-cluster-name tkg-cluster-01
--tanzu-kubernetes-cluster-namespace tkg-cluster-ns
--vsphere-username operator@example.com
```

Esta acción crea un archivo de configuración con el token web de JSON (JSON Web Token, JWT) para autenticarse en la API de Kubernetes.

3 Para autenticarse, introduzca la contraseña de vCenter Single Sign-On.

Si la operación se realiza correctamente, aparecerá el mensaje `Logged in successfully` y podrá ejecutar comandos de kubectl en el clúster. Si el comando devuelve el error `Error from server (Forbidden)`, es posible que no tenga los permisos necesarios.

4 Para obtener una lista de los contextos a su disposición, ejecute el siguiente comando:

```
kubectl config get-contexts
```

Este comando muestra los contextos de configuración a los que puede acceder. Debe ver un contexto de configuración para el clúster de destino como, por ejemplo, `tkg-cluster-01`.

5 Para usar el contexto del clúster de destino, ejecute el siguiente comando:

```
kubectl config use-context CLUSTER-NAME
```

6 Para enumerar los nodos de clúster, ejecute el siguiente comando:

```
kubectl get nodes
```

Verá el plano de control y los nodos de trabajo de este clúster.

7 Para enumerar todos los pods de clúster, ejecute el siguiente comando:

```
kubectl get pods -A
```

Verá todos los pods de este clúster en todos los espacios de nombres de Kubernetes a los que puede acceder. Si no implementó ninguna carga de trabajo, no verá ningún pod en el espacio de nombres predeterminado.

# Conceder a los desarrolladores acceso de vCenter SSO a los clústeres de Servicio TKG

Los usuarios desarrolladores y los grupos de desarrollo son los usuarios de destino de los clústeres de Servicio TKG. Una vez que se aprovisiona un clúster de Servicio TKG, es posible otorgar acceso de desarrollador mediante la autenticación de vCenter Single Sign-On o a través de un proveedor de identidad externo compatible.

## Autenticación para desarrolladores

Un administrador de clústeres puede otorgar acceso al clúster a otros usuarios, como desarrolladores. Los desarrolladores pueden implementar pods en clústeres directamente mediante sus cuentas de usuario o de forma indirecta a través de cuentas de servicio.

- Para la autenticación de la cuenta de usuario, los clústeres de Servicio TKG admiten usuarios y grupos de vCenter Single Sign-On. El usuario o el grupo pueden ser locales para la instancia de vCenter Server o sincronizarse desde un servidor de directorio compatible.
- Los usuarios y grupos de OIDC externos se asignan directamente a funciones del espacio de nombres de vSphere.
- Para la autenticación de la cuenta de servicio, puede utilizar tokens de servicio. Para obtener más información, consulte la documentación de Kubernetes.

## Agregar usuarios desarrolladores a un clúster

Para conceder acceso al clúster a desarrolladores, haga lo siguiente:

1 Defina una función o ClusterRole para el usuario o el grupo y aplíquelos al clúster. Para obtener más información, consulte la documentación de Kubernetes.

2 Cree un RoleBinding o ClusterRoleBinding para el usuario o grupo y aplíquelo al clúster. Vea el ejemplo siguiente:

### Ejemplo de RoleBinding

Para conceder acceso a un usuario o grupo de vCenter Single Sign-On, el asunto en RoleBinding debe contener uno de los siguientes valores para el parámetro `name`.

**Tabla 4-2. Campos de usuario y grupo admitidos**

| Campo | Descripción |
| --- | --- |
| `sso:`*`USER-NAME@DOMAIN`* | Por ejemplo, un nombre de usuario local, como `sso:joe@vsphere.local`. |
| `sso:`*`GROUP-NAME@DOMAIN`* | Por ejemplo, un nombre de grupo de un servidor de directorio integrado con la instancia de vCenter Server, como `sso:devs@ldap.example.com`. |

El siguiente ejemplo de RoleBinding enlaza el usuario local de vCenter Single Sign-On, llamado Joe, al objeto ClusterRole predeterminado denominado `edit`. Esta función permite el acceso de lectura/escritura a la mayoría de los objetos en un espacio de nombres, en este caso, el espacio de nombres `default`.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-cluster-user-joe
  namespace: default
roleRef:
  kind: ClusterRole
  name: edit                             #Default ClusterRole
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: sso:joe@vsphere.local            #sso:<username>@<domain>
  apiGroup: rbac.authorization.k8s.io
```

## Conectarse a Supervisor mediante la CLI de Tanzu y la autenticación de vCenter SSO

Siga estos pasos para conectarse a Supervisor mediante la CLI de Tanzu y autenticarse como usuario de vCenter Single Sign-On.

### Requisitos previos

Complete los siguientes requisitos previos.

1 Instale y configure la instancia de Herramientas de la CLI de Kubernetes para vSphere. Consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

2 Instale e inicialice la CLI de Tanzu. Consulte Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG.

### Conectarse a Supervisor mediante la CLI de Tanzu y vCenter SSO

Complete los siguientes pasos.

1 Conéctese a Supervisor como usuario de vCenter SSO.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS --vsphere-username
VCENTER-SSO-USER
```

Esta acción rellena kubeconfig con el contexto de Supervisor que utiliza la CLI de Tanzu.

2 Cambie el contexto al espacio de nombres de vSphere para Supervisor.

```
kubectl config get-contexts
```

```
kubectl config use-context <SUPERVISOR-CONTROL-PLANE-IP-ADDRESS>
```

3 Inicie sesión en Supervisor mediante la CLI de Tanzu y vCenter SSO.

```
tanzu context create context_name --kubeconfig ~/.kube/config --kubecontext SUPERVISOR-
CONTROL-PLANE-IP-ADDRESS
```

Donde:

- `context_name` es el nombre definido por el usuario para este contexto, como "supervisor".
- `--kubeconfig ~/.kube/config` es la ruta de acceso al archivo `kubeconfig` local, que se establece de forma predeterminada para `~/.kube/config` y se establece mediante la variable de entorno `KUBECONFIG`, y que incluye el contexto de configuración de Supervisor para el usuario de vCenter SSO.
- `--kubecontext SUPERVISOR-CONTROL-PLANE-IP-ADDRESS` es el contexto del Supervisor, que es el mismo que `SUPERVISOR_IP`, como `10.179.144.55`.

4 Ejecute los comandos de la CLI de Tanzu y compruebe la conectividad.

```
tanzu plugin list
```

```
tanzu cluster list -n VSPHERE-NS-FOR-TKG
```

## Conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo

Puede conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo y el servicio de autenticación Pinniped para Kubernetes.

# Configurar un IDP externo para usarlo con clústeres de servicio TKG

Es posible configurar Supervisor con cualquier proveedor de identidad (Identity Provider, IDP) compatible con OIDC, como Okta. Para completar la integración, configure el IDP con la URL de devolución de llamada para Supervisor.

## Proveedores de OIDC externos compatibles

Es posible configurar el Supervisor con cualquier proveedor de identidad compatible con OIDC. La tabla enumera los más comunes y proporciona vínculos a las instrucciones de configuración.

| IDP externo | Configuración |
| --- | --- |
| Okta | Ejemplo de configuración de OIDC mediante Okta<br>Consulte también Configurar Okta como proveedor de OIDC para Pinniped. |
| Workspace ONE | Configurar Workspace ONE Access como proveedor de OIDC para Pinniped |
| Dex | Configurar Dex como proveedor de OIDC para Pinniped |
| GitLab | Configurar GitLab como proveedor de OIDC para Pinniped |
| Google OAuth | Uso de Google OAuth 2 |

## Configurar el IDP con la URL de devolución de llamada para el Supervisor

El Supervisor actúa como un cliente de OAuth 2.0 para el proveedor de identidad externo. La URL de devolución de llamada del Supervisor es la URL de redireccionamiento que se utiliza para configurar el IDP externo. La URL de devolución de llamada tiene el formato *https://SUPERVISOR-VIP/wcp/pinniped/callback*.

**Nota** Al realizar el registro de IDP, es posible que la URL de devolución de llamada se llame "URL de redireccionamiento" en el proveedor de OIDC que está configurando.

Al configurar el proveedor de identidad externo para usarlo con TKG en Supervisor, proporcione al proveedor de identidad externo la **URL de devolución de llamada** que se proporciona en vCenter Server en la pantalla **Administración de cargas de trabajo > Supervisores > Configurar > Proveedores de identidad**.

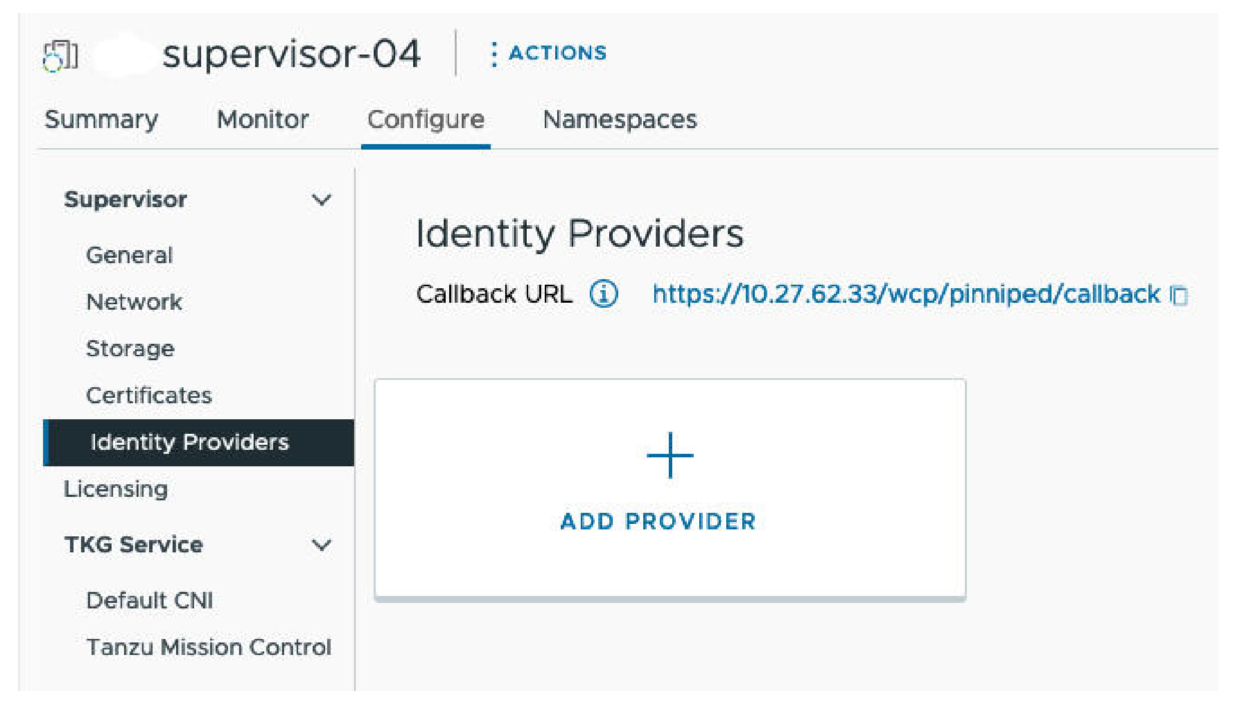
## Ejemplo de configuración de OIDC mediante Okta

Okta permite a los usuarios iniciar sesión en las aplicaciones mediante el protocolo OpenID Connect. Cuando se configura Okta como proveedor de identidad externo para Tanzu Kubernetes Grid en Supervisor, los pods Pinniped en Supervisor y en clústeres de Tanzu Kubernetes Grid controlan el acceso de los usuarios a los espacios de nombres de vSphere y los clústeres de cargas de trabajo.

1 Copie la URL de devolución de llamada del proveedor de identidad para el que necesita crear una conexión de OIDC entre Okta y vCenter Server.

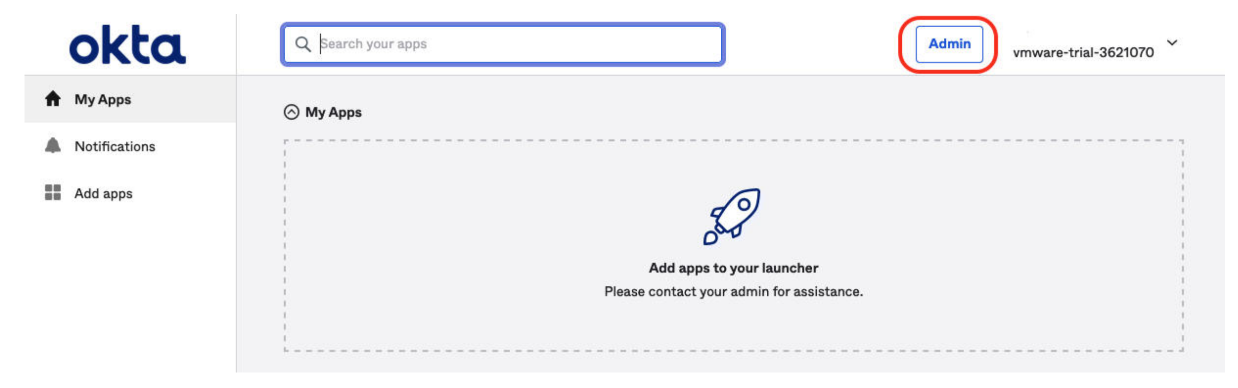
Con vSphere Client, obtenga la URL de devolución de llamada del proveedor de identidad en **Administración de cargas de trabajo > Supervisores > Configurar > Proveedores de identidad**. Copie esta URL en una ubicación temporal.

Figura 4-2. URL de devolución de llamada de IDP



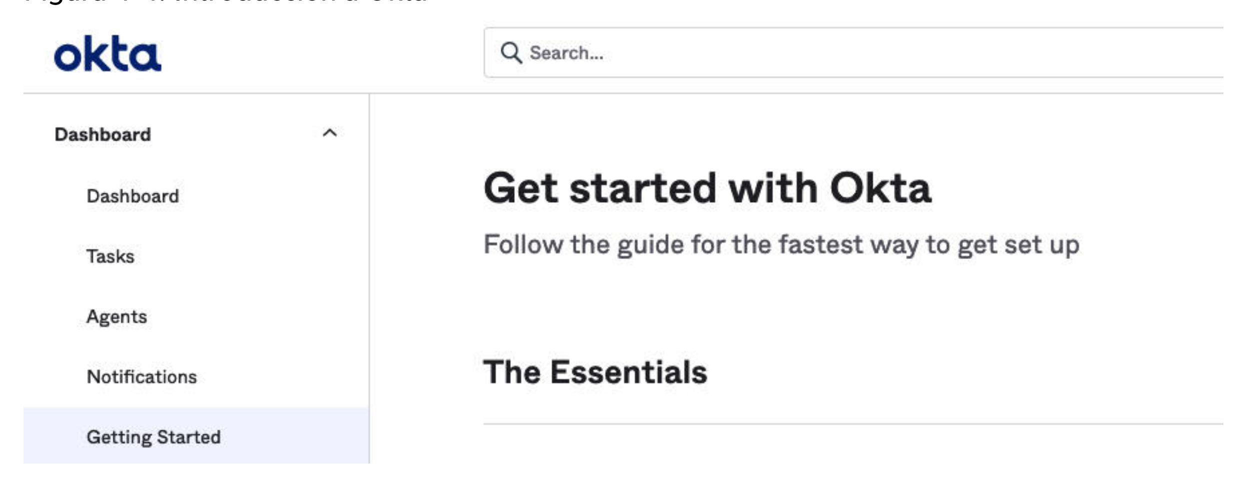

2 Inicie sesión en la cuenta de Okta de su organización o cree una cuenta de prueba en https://www.okta.com/. Haga clic en el botón **Admin** para abrir la consola administrativa de Okta.

Figura 4-3. Consola administrativa de Okta




3 En la página de introducción de la consola administrativa, desplácese hasta **Applications (Aplicaciones) > Applications (Aplicaciones)**.

Figura 4-4. Introducción a Okta




4 Seleccione la opción para **crear una integración de aplicaciones**.

Figura 4-5. Crear integración de aplicaciones de Okta

Applications

Create App Integration | Browse App Catalog | Assign Users to App | More

5 Cree la nueva integración de aplicaciones.

- Establezca el método de inicio de sesión en **OIDC - OpenID Connect**.
- Establezca el tipo de aplicación en una **aplicación web**.

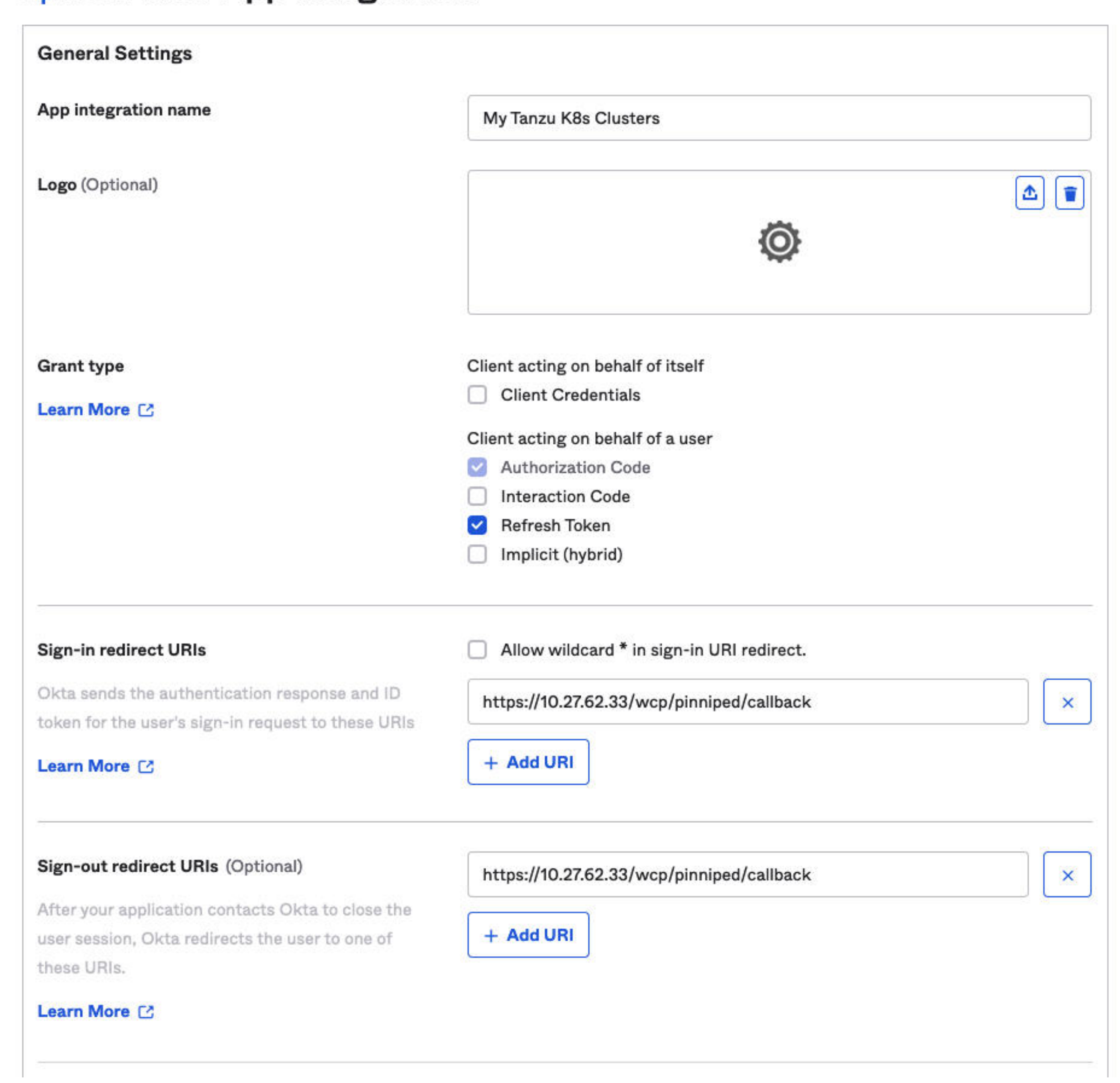
Figura 4-6. Método de inicio de sesión de Okta y tipo de aplicación



6 Configure los detalles de integración de la aplicación web de Okta.

- Proporcione un **nombre para la integración de aplicaciones**, que sea una cadena definida por el usuario.
- Especifique el **tipo de concesión**: se debe seleccionar **Authorization Code** (Código de autorización) y **Refresh Token** (Token de referencia).
- URI de redirección de inicio de sesión: introduzca la URL de devolución de llamada del proveedor de identidad que copió de Supervisor (vea el paso 1), como https://10.27.62.33/wcp/pinnipend/callback.
- URI de redirección de cierre de sesión: introduzca la URL de devolución de llamada del proveedor de identidad que copió de Supervisor (vea el paso 1), como https://10.27.62.33/wcp/pinnipend/callback.

Figura 4-7. Detalles de integración de aplicaciones web de Okta




7 Configure el control de acceso de los usuarios.

En la sección **Assignments (Asignaciones) > Controlled access (Acceso controlado)**, tiene la opción de controlar si lo desea cuáles de los usuarios de Okta que existen en su organización pueden acceder a los clústeres de Tanzu Kubernetes Grid. En el ejemplo, permita que todos los definidos en la organización tengan acceso.

Figura 4-8. Control de acceso de Okta

Trusted Origins

Base URIs (Optional)

Required if you plan to self-host the Okta Sign-In Widget. With a Trusted Origin set, the Sign-In Widget can make calls to the authentication API from this domain.

Learn More

+ Add URI

Assignments

Controlled access

Select whether to assign the app integration to everyone in your org, only selected group(s), or skip assignment until after app creation.

Allow everyone in your organization to access

Limit access to selected groups

Skip group assignment for now

Enable immediate access (Recommended)

Recommended if you want to grant access to everyone without pre-assigning your app to users and use Okta only for authentication.

Enable immediate access with **Federation Broker Mode**

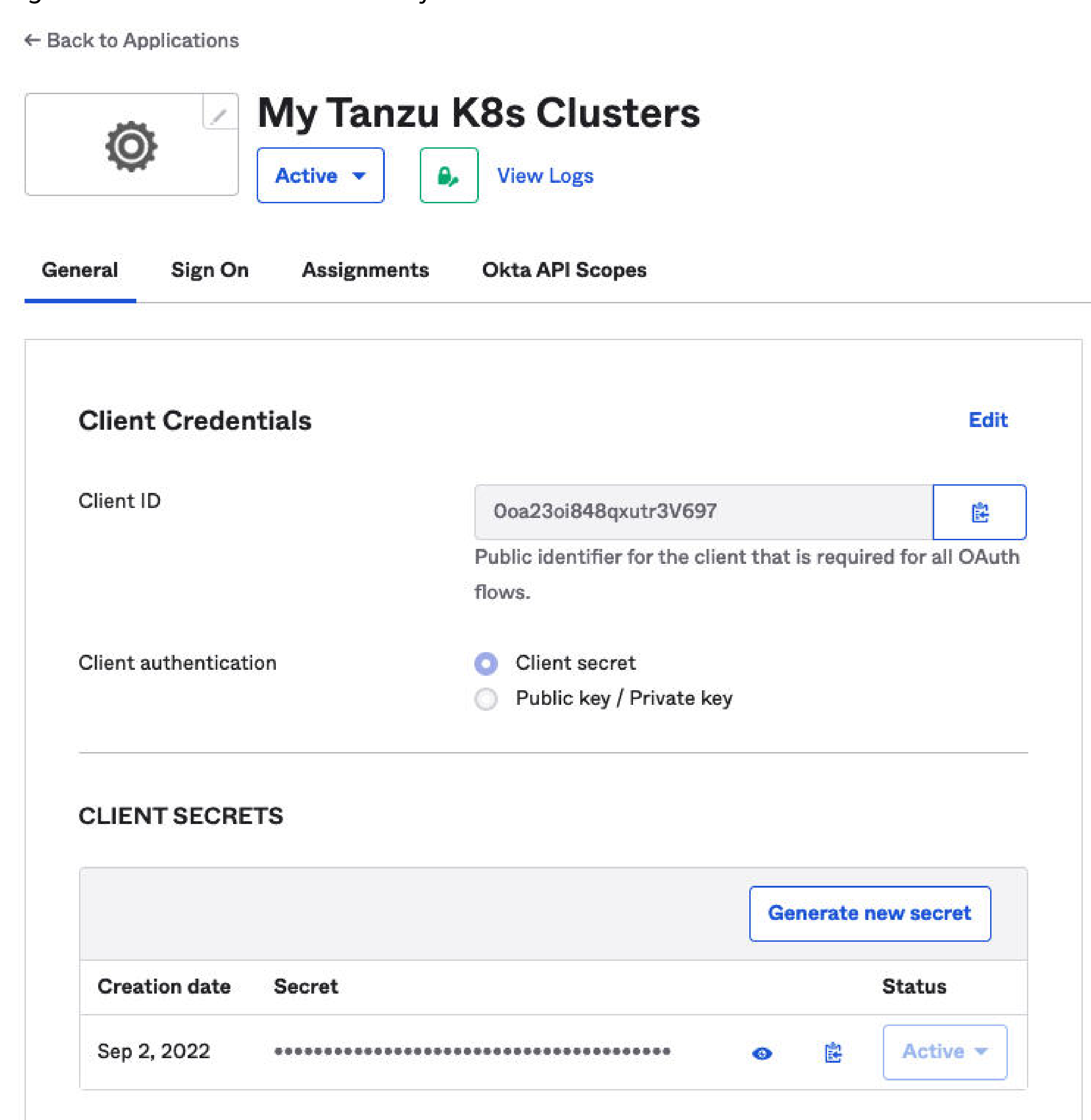
To ensure optimal app performance at scale, Okta End User Dashboard and provisioning features are disabled. Learn more about Federation Broker Mode.

Save Cancel

8 Haga clic en **Save** (Guardar) y copie el **identificador de cliente** y el **secreto de cliente** que se devuelven.

Para guardar la configuración de Okta, la consola administrativa le proporciona un **ID de cliente** y un **secreto de cliente**. Copie ambos datos porque los va a necesitar para configurar Supervisor con un proveedor de identidad externo.

Figura 4-9. Identificador de cliente y secreto de OIDC




9 Configure el token de ID de OpenID Connect.

Haga clic en la pestaña **Sign On**. En la sección **OpenID Connect ID Token** (Token de ID de OpenID Connect), haga clic en el vínculo Edit (Editar), rellene el filtro **Groups claim type** (Tipo de notificación de grupos) y **guarde** la configuración.

Por ejemplo, si desea que el nombre de notificación "groups" coincida en todos los grupos, seleccione **groups > Matches regex >** *.

Figura 4-10. Token de ID de OpenID Connect

**OpenID Connect ID Token** Cancel

| | |
|---|---|
| Issuer | Dynamic (based on request domain) |
| Audience | 0oa23o0aei0TXYuG3697 |
| Claims | Claims for this token include all user attributes on the app profile. |
| Groups claim type | Filter |
| Groups claim filter | groups · Matches regex · * <br> Using Groups Claim |

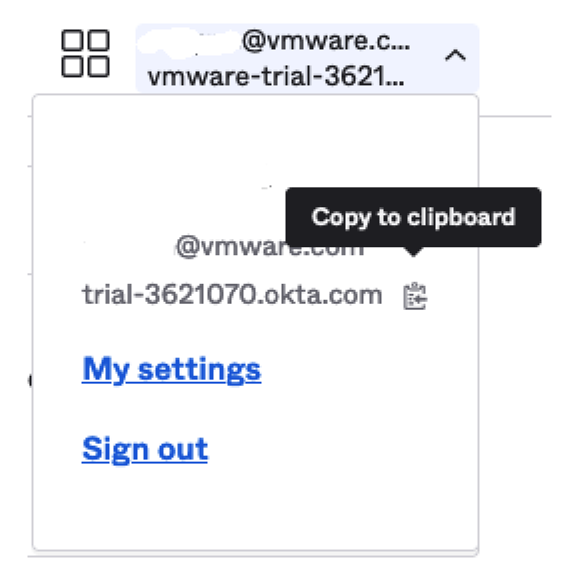
Save Cancel

10 Copie la **URL del emisor**.

Para configurar el Supervisor, necesita la **URL del emisor**, además del **ID de cliente** y el **secreto de cliente**.

Copie la **URL del emisor** desde la consola administrativa de Okta.

Figura 4-11. URL del emisor de Okta



# Registrar un IDP externo con Supervisor

Para conectarse a clústeres de Tanzu Kubernetes Grid 2.0 en Supervisor mediante la CLI de Tanzu, registre el proveedor de OIDC con Supervisor.

## Requisitos previos

Antes de registrar un proveedor de ODIC externo con Supervisor, complete los siguientes requisitos previos:

- Habilite Administración de cargas de trabajo e implemente una instancia de Supervisor. Consulte Ejecutar clústeres de TKG 2.0 en Supervisor.
- Configure un proveedor de identidad externo OpenID Connect con la URL de devolución de llamada de Supervisor. Consulte Configurar un IDP externo para usarlo con clústeres de servicio TKG.
- Obtenga el identificador de cliente, el secreto de cliente y la URL del emisor del IDP externo. Consulte Configurar un IDP externo para usarlo con clústeres de servicio TKG.

## Registrar un IDP externo con Supervisor

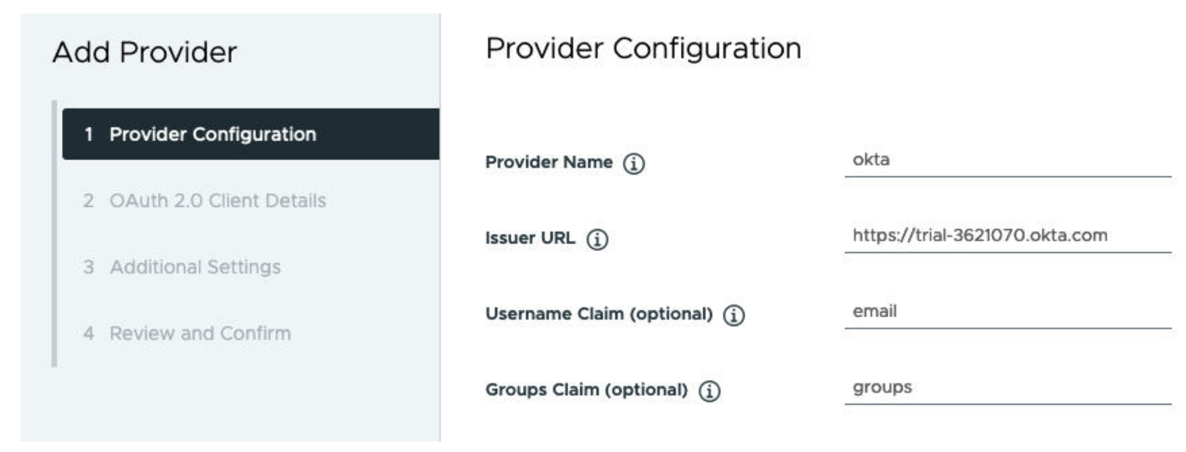
Supervisor ejecuta como pods los componentes Supervisor de Pinniped y Conserje de Pinniped. Los clústeres de Tanzu Kubernetes Grid solo ejecutan como pods el componente Conserje de Pinniped. Para obtener más información sobre estos componentes y cómo interactúan, consulte la documentación del servició de autenticación Pinniped.

Una vez que se registra un proveedor de identidad externo con Supervisor, el sistema actualiza los pods Supervisor y Conserje de Pinniped en Supervisor y los pods Conserje de Pinniped en clústeres de Tanzu Kubernetes Grid. Todos los clústeres de Tanzu Kubernetes Grid que se ejecutan en esa instancia de Supervisor se configuran automáticamente con ese mismo proveedor de identidad externo.

Para registrar un proveedor de ODIC externo con Supervisor, realice el siguiente procedimiento:

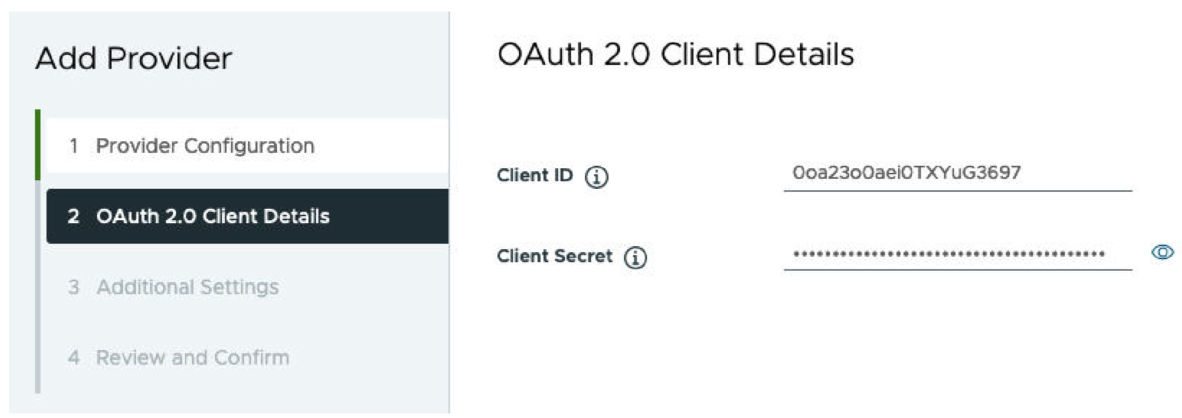
1. Inicie sesión en vCenter Server mediante vSphere Client.
2. Seleccione **Administración de cargas de trabajo > Supervisores > Configurar > Proveedores de identidad**.
3. Haga clic en el signo más para comenzar el proceso de registro.
4. Configure el proveedor. Consulte Configuración del proveedor de OIDC.

Figura 4-12. Configuración del proveedor de OIDC




5 Configure los detalles del cliente de OAuth 2.0. Consulte Detalles del cliente de OAuth 2.0.

Figura 4-13. Detalles del cliente de OAuth 2.0




6 Configure los ajustes adicionales. Consulte Configuración adicional.

7 Confirme la configuración del proveedor.

Figura 4-14. Confirmar configuración del proveedor

Add Provider

1 Provider Configuration

2 OAuth 2.0 Client Details

3 Additional Settings

4 Review and Confirm

Review and Confirm

**Provider Configuration**

| | |
|---|---|
| Provider Name | okta |
| Issuer URL | https://trial-3621070.okta.com |
| Username Claim | email |
| Groups Claim | groups |

**OAuth 2.0 Client Details**

| | |
|---|---|
| Client ID | 0oa23o0aei0TXYuG3697 |
| Client Secret | **************************************** |

**Additional Settings**

| | |
|---|---|
| Additional Scopes | -- |
| Certificate Authority Data | -- |
| Additional Authorize Parameters | -- |

CANCEL BACK FINISH

8 Haga clic en **Finalizar** para completar el registro del proveedor de OIDC.

## Configuración del proveedor de OIDC

Consulte los siguientes detalles de configuración del proveedor al registrar un proveedor de OIDC externo con Supervisor.

Tabla 4-3. Configuración del proveedor de OIDC

| Campo | Importancia | Descripción |
|---|---|---|
| Nombre del proveedor | Obligatorio | El nombre definido por el usuario para el proveedor de identidad externo. |
| URL del emisor | Obligatorio | La URL del proveedor de identidad que emite tokens. La URL de detección de OIDC se deriva de la URL del emisor.<br>Por ejemplo, para Okta, la URL del emisor puede ser similar a la siguiente y puede obtenerse en la consola de administración: *https://trial-4359939-admin.okta.com*. |

Tabla 4-3. Configuración del proveedor de OIDC (continuación)

| Campo | Importancia | Descripción |
| --- | --- | --- |
| Notificación de nombre de usuario | Opcional | La notificación del token de identificador del proveedor de identidad ascendente o del endpoint de información de usuario que se va a inspeccionar para obtener el nombre de usuario para el usuario especificado. Si deja este campo vacío, la URL del emisor ascendente se concatena con la notificación "sub" para generar el nombre de usuario que se utilizará con Kubernetes.<br>Este campo especifica lo que Pinniped debe examinar en el token de identificador ascendente para determinar la autenticación. Si no se proporciona, la identidad del usuario llevará el formato *https://IDP-ISSUER?sub=UUID*. |
| Notificación de grupos | Opcional | La notificación del token de identificador del proveedor de identidad ascendente o del endpoint de información de usuario que se va a inspeccionar para obtener los grupos para el usuario especificado. Si deja este campo vacío, no se utilizarán grupos del proveedor de identidad ascendente.<br>El campo de notificaciones de grupos indica a Pinniped qué se debe buscar en el token de ID ascendente para autenticar la identidad del usuario. |

## Detalles del cliente de OAuth 2.0

Consulte los siguientes detalles del cliente OAuth 2.0 de proveedor al registrar un proveedor de OIDC externo con Supervisor.

Tabla 4-4. Detalles del cliente de OAuth 2.0

| Detalles del cliente de OAuth 2.0 | Importancia | Descripción |
| --- | --- | --- |
| Identificador de cliente | Obligatorio | Identificador de cliente del IDP externo |
| Secreto del cliente | Obligatorio | Secreto de cliente del IDP externo |

## Configuración adicional

Consulte la siguiente configuración adicional al registrar un proveedor de OIDC externo con Supervisor.

Tabla 4-5. Configuración adicional

| Configuración | Importancia | Descripción |
| --- | --- | --- |
| Ámbitos adicionales | Opcional | Los ámbitos adicionales que se solicitarán en los tokens. |
| Datos de la entidad de certificación | Opcional | Datos de la entidad de certificación TLS para una conexión IDP externa segura |
| Parámetros de autorización adicionales | Opcional | Parámetros adicionales durante la solicitud de autorización de OAuth2 |

# Configurar permisos de espacio de nombres de vSphere para usuarios y grupos del proveedor de identidad externo

A fin de configurar el acceso de los usuarios de OIDC al clúster de TKG 2.0, configure el espacio de nombres de vSphere con permisos de función para los usuarios y grupos del proveedor de identidad externo.

## Configurar permisos de espacio de nombres de vSphere para usuarios y grupos del proveedor de identidad externo

Un clúster de TKG 2.0 en Supervisor se aprovisiona en un espacio de nombres de vSphere. Después de registrar un proveedor de OIDC externo con Supervisor, configure el espacio de nombres de vSphere con permisos de función para usuarios y grupos de proveedores de OIDC externos. Esta acción crea los enlaces de función para el proveedor de OIDC externo en cada clúster de TKG 2.0 en espacio de nombres de vSphere. Si el espacio de nombres de vSphere ya existe, se actualizan los enlaces de funciones.

**Nota** Una vez registrado un IDP externo en un Supervisor, todos los clústeres de TKG 2.0 que se crearon en ese Supervisor se configurarán automáticamente con el IDP externo a través de los componentes de Pinniped.

1 Registre un proveedor de identidad externo con Supervisor.

   Consulte Registrar un IDP externo con Supervisor.

2 Cree un espacio de nombres de vSphere para uno o varios clústeres de TKG o seleccione un espacio de nombres de vSphere existente.

   Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

3 Configure usuarios y funciones para el espacio de nombres de vSphere.

   Seleccione el proveedor de OIDC externo como origen de identidad, agregue usuarios y asigne funciones.

   a Seleccione el espacio de nombres de vSphere.

   b Seleccione **Permisos > Agregar permisos**.

c **Origen de identidad**: seleccione el proveedor de identidad externo que se registró con Supervisor.

El **nombre de proveedor** que utilizó para registrar el IDP externo debe aparecer en el menú desplegable. Si no es así, compruebe la configuración.

d **Búsqueda de usuarios/grupos**: escriba el nombre del usuario o grupo. La entrada de texto es una cadena de formato libre.

Los usuarios y los grupos de un proveedor de identidad externo no se sincronizan con vCenter Server y no se pueden seleccionar. Debe escribir en el valor de la cadena, por lo general, una dirección de correo electrónico. No hay ningún prefijo, por lo que puede escribir "jdoe@acme.com", por ejemplo.

e **Función**: para asignar una función al usuario o al grupo, seleccione la función, ya sea **Puede ver** o **Puede editar**.

**Nota** La función Propietario no está disponible para poder usarla con un proveedor de identidad externo.

4 Complete la configuración del espacio de nombres de vSphere.

Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

## Conectarse a Supervisor mediante la CLI de Tanzu y un IDP externo

Siga estos pasos para conectarse a Supervisor mediante la CLI de Tanzu.

### Requisitos previos

Complete los siguientes requisitos previos.

1 Registre con Supervisor un proveedor de identidad externo compatible con OIDC. Consulte Registrar un IDP externo con Supervisor.

2 Conceda a los usuarios y grupos de OIDC acceso a un espacio de nombres de vSphere. Consulte Configurar permisos de espacio de nombres de vSphere para usuarios y grupos del proveedor de identidad externo.

### Conectarse al Supervisor mediante la CLI de Tanzu

Complete los siguientes pasos.

**Nota** Si utiliza el servicio TKG 3.1 o versiones posteriores, descargue el complemento de CLI de autenticación de Pinniped con la misma versión del servicio de TKG y descargue el complemento de CLI imgpkg con la versión más reciente. Para obtener más información, consulte la documentación de producto de la CLI de Tanzu.

1 Instale e inicialice la CLI de Tanzu. Consulte Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG.

2 Ejecute el siguiente comando para conectarse a Supervisor.

```
tanzu context create context_name --endpoint https://10.73.27.32
```

Donde:

- El valor `context_name` es el nombre de un OIDC al que se concede acceso
- El valor `--endpoint` es la dirección IP del plano de control de Supervisor

**Nota** Para solucionar problemas, anexe `--stderr-only` al comando, por ejemplo: `tanzu login --endpoint https://IP --name USER --stderr-only`.

3 Cuando se emita el reto, visite el vínculo con su navegador.

**Figura 4-15. Inicio de sesión de la CLI de Tanzu**

**Finish your login**

To finish logging in, paste this authorization code into your command-line session:




4 Copie y pegue el código de autorización en la CLI de Tanzu.

```
tanzu context create context_name --endpoint https://10.73.27.32

Detected a vSphere Supervisor being used
Log in by visiting this link:
...
https://10.27.62.33/wcp/pinniped/oauth2/authorize?..
...
Optionally, paste your authorization code:
G2TcS145Q4e6A1YKf743n3BJlfQAQ_UdjXy38TtEEIo.ju4QV3PTsUvOigVUtQllZ7AJFU0YnjuLHTRVoNxvdZc
...
✔ successfully logged in to management cluster using the kubeconfig oidc-user
Checking for required plugins...
All required plugins are already installed and up-to-date
```

5 Una vez autenticado, puede utilizar la CLI de Tanzu para aprovisionar un clúster de TKG en el espacio de nombres de vSphere de destino al que tiene acceso. Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante la CLI de Tanzu.

## Conectarse a un clúster de TKG como usuario de OIDC con la CLI de Tanzu

Puede conectarse a un clúster de TKG mediante la CLI de Tanzu y autenticarse con su proveedor de OIDC.

## Requisitos previos

En estas instrucciones se supone que Supervisor está configurado con un proveedor de identidad externo (IDP) compatible, que usted (un usuario de desarrollo y operaciones) se conectó a Supervisor mediante la CLI de Tanzu y que se aprovisionó un clúster de TKG. Consulte los siguientes temas según sea necesario:

- Conectarse a clústeres de TKG en Supervisor mediante un proveedor de identidad externo
- Capítulo 7 Aprovisionar clústeres del servicio de TKG

## Flujo de trabajo de usuario de desarrollo y operaciones

Como usuario de desarrollo y operaciones con permisos de edición en la instancia de espacio de nombres de vSphere de destino, utilice la CLI de Tanzu para generar un archivo `kubeconfig` que se pueda compartir y que, a continuación, se distribuya a los usuarios de los clústeres de TKG. En Kubernetes, un contexto de configuración incluye un clúster, un espacio de nombres y un usuario. Puede ver el contexto del clúster en el archivo `.kube/config`. Generalmente, este archivo se denomina `kubeconfig`.

**Nota** Estos pasos los debe realizar un usuario de desarrollo y operaciones con permisos de edición en la instancia de espacio de nombres de vSphere de destino.

1 Compruebe que el contexto de la CLI de Tanzu esté establecido en Supervisor.

  Consulte Conectarse a Supervisor mediante la CLI de Tanzu y un IDP externo.

2 Compruebe que el administrador de vSphere haya configurado permisos de usuario para la instancia de espacio de nombres de vSphere de destino.

  Los usuarios y grupos de OIDC externos se asignan directamente a funciones del espacio de nombres de vSphere. Los usuarios del clúster deben agregarse a espacio de nombres de vSphere antes de generar el kubeconfig que se va a compartir.

  Consulte Configurar permisos de espacio de nombres de vSphere para usuarios y grupos del proveedor de identidad externo.

3 Enumere los clústeres de TKG aprovisionados en la instancia de espacio de nombres de vSphere de destino.

```
tanzu cluster list --namespace VSPHERE-NAMESPACE
```

4 Genere un archivo kubeconfig que se pueda compartir para el clúster de TKG de destino.

```
tanzu cluster kubeconfig get CLUSTER-NAME --namespace=NAMESPACE
```

5 Distribuya el archivo kubeconfig compartido a los usuarios del clúster para que puedan iniciar sesión en el clúster de TKG.

### Flujo de trabajo de usuario del clúster

Complete estos pasos para iniciar sesión en un clúster de TKG como usuario de clúster.

1 Obtenga el archivo kubeconfig del usuario de desarrollo y operaciones.

2 Inicie sesión en el clúster de TKGS mediante el archivo kubeconfig y kubectl.

```
kubectl --kubeconfig
```

3 Complete el proceso de autenticación del navegador.

   a Cuando se emita el reto, visite el vínculo con su navegador.

   b Copie y pegue el código de autorización en la CLI.

4 Utilice `kubectl` para interactuar con el clúster.

# Conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes

Puede conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes para ayudar en la administración y la solución de problemas de los clústeres de Servicio TKG.

## Conectarse al plano de control del clúster de Servicio TKG como administrador de Kubernetes

Puede conectarse al plano de control del clúster de Servicio TKG como el usuario `kubernetes-admin` para realizar tareas administrativas y solucionar problemas del clúster.

Un archivo `kubeconfig` válido para un clúster de Tanzu Kubernetes aprovisionado está disponible en Supervisor como un objeto secreto denominado *`TKG-CLUSTER-NAME`*`-kubeconfig`. Puede utilizar este secreto para conectarse al plano de control del clúster como el usuario `kubernetes-admin`.

#### Procedimiento

**1** Conéctese a Supervisor.

**2** Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
kubectl config use-context VSPHERE-NAMESPACE
```

**3** Vea los objetos secretos en el espacio de nombres.

```
kubectl get secrets
```

El secreto se denomina *TKG-CLUSTER-NAME*-kubeconfig.

```
kubectl config use-context tkg-cluster-ns
Switched to context "tkg-cluster-ns".
ubuntu@ubuntu:~$ kubectl get secrets
NAME                                 TYPE                                  DATA   AGE
...
tkg-cluster-1-kubeconfig            Opaque                                1      23h
...
```

4 Ejecute el siguiente comando para descodificar el secreto.

El secreto está codificado en Base64. Para descodificarlo: en Linux, utilice `base64 --decode` (o `base64 -d`); en MacOS, utilice `base64 --Decode` (o `base64 -D`); en Windows, utilice una herramienta en línea.

```
kubectl get secret TKG-CLUSTER-NAME-kubeconfig -o jsonpath='{.data.value}' | base64 -d >
tkgs-cluster-kubeconfig-admin
```

Este comando descodifica el secreto y lo escribe en un archivo local denominado `tkgs-cluster-kubeconfig-admin`. Utilice el comando `cat` para comprobar el contenido del archivo.

5 Conéctese al clúster de TKG como administrador de Kubernetes mediante el archivo `tkg-cluster-kubeconfig-admin` decodificado.

Existen dos opciones para realizar esta acción:

| Opción | Descripción |
| --- | --- |
| **--kubeconfig <path\to\kubeconfig>** | Utilice la marca `--kubeconfig` y la ruta de acceso al archivo kubeconfig local. Por ejemplo, si suponemos que el archivo kubeconfig se encuentra en el mismo directorio en el que se ejecuta el comando: `kubectl --kubeconfig tkg-cluster-kubeconfig-admin get nodes` |
| **KUBECONFIG** | Establezca la variable de entorno KUBECONFIG para que apunte al archivo kubeconfig descodificado y ejecute kubectl, como `kubectl get nodes`. |

Debería ver los nodos en el clúster.

6 Si es un usuario de desarrollo y operaciones con permisos de edición en espacio de nombres de vSphere y desea iniciar sesión en un clúster de TKG como usuario administrador mediante la CLI de Tanzu, ejecute el siguiente comando:

```
tanzu cluster kubeconfig get CLUSTER-NAME --admin
```

Este comando generará un archivo `kubeconfig` que contiene la clave cert/private para kubernetes-control-plane (que omite toda la autorización). A continuación, puede iniciar sesión en el clúster mediante este archivo `kubeconfig`. Consulte #unique_36.

# Conectarse mediante SSH a nodos de clúster de Servicio TKG como usuario del sistema con una clave privada

Puede conectarse mediante SSH a un nodo de clúster de TKG como `vmware-system-user` con una clave privada.

Puede conectarse mediante SSH a cualquier nodo del clúster de TKG como usuario de `vmware-system-user`. El secreto que contiene la clave privada SSH se denomina `CLUSTER-NAME-ssh`. Consulte Obtener secretos de clúster de TKG mediante Kubectl.

Para conectarse a un nodo del clúster de TKG a través de SSH mediante una clave privada, cree un pod de vSphere de Jump Box en el Supervisor.

### Requisitos previos

En esta tarea se aprovisiona un pod de vSphere como host de salto para la conectividad SSH. pods de vSphere requiere redes NSX para Supervisor. Si utiliza redes de vDS para Supervisor, consulte Conectarse mediante SSH a nodos de clúster de Servicio TKG como usuario del sistema con una contraseña.

### Procedimiento

1 Conéctese a Supervisor.

Consulte Conectarse a Supervisor como usuario de vCenter Single Sign-On con kubectl.

2 Cree una variable de entorno denominada **`NAMESPACE`** cuyo valor sea el nombre del espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
export NAMESPACE=VSPHERE-NAMESPACE
```

3 Cambie el contexto a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster de Tanzu Kubernetes.

```
kubectl config use-context $NAMESPACE
```

4 Consulte el objeto secreto *`TKG-CLUSTER-NAME`*`-ssh`.

```
kubectl get secrets
```

5 Cree un secreto de credencial para el registro de Docker Hub.

De forma predeterminada, la imagen utilizada para crear el pod de vSphere (Photon OS) se extrae de Docker Hub. Es posible que necesite un secreto de credencial para extraer correctamente la imagen. Consulte Crear secreto de credencial de registro privado.

6 Cree una instancia de pod de vSphere mediante las siguientes especificaciones `jumpbox.yaml`.

Reemplace el valor de `namespace` `YOUR-NAMESPACE` por el espacio de nombres de vSphere en el que se aprovisiona el clúster de destino. Reemplace el valor de `secretName` `YOUR-CLUSTER-NAME`-ssh con el nombre del clúster de destino.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox
  namespace: YOUR-NAMESPACE     #REPLACE
spec:
  containers:
  - image: "photon:3.0"
    name: jumpbox
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "yum install -y openssh-server; mkdir /root/.ssh; cp /root/ssh/ssh-privatekey /
root/.ssh/id_rsa; chmod 600 /root/.ssh/id_rsa; while true; do sleep 30; done;" ]
    volumeMounts:
      - mountPath: "/root/ssh"
        name: ssh-key
        readOnly: true
    resources:
      requests:
        memory: 2Gi
  volumes:
    - name: ssh-key
      secret:
        secretName: YOUR-CLUSTER-NAME-ssh     #REPLACE
  imagePullSecrets:
    - name: regcred
```

7 Implemente el pod aplicando la especificación `jumpbox.yaml`.

```
kubectl apply -f jumpbox.yaml
```

```
pod/jumpbox created
```

8 Compruebe que el pod se esté ejecutando.

```
kubectl get pods
```

```
NAME      READY   STATUS    RESTARTS   AGE
jumpbox   1/1     Running   0          3h9m
```

**Nota** También debe ver el pod de Jump Box en vCenter en el espacio de nombres de vSphere.

9 Cree una variable de entorno con la dirección IP del nodo de clúster de destino ejecutando el siguiente conjunto de comandos.

a Obtenga el nombre de la máquina virtual de destino.

```
kubectl get virtualmachines
```

b Cree la variable de entorno `VMNAME` cuyo valor sea el nombre del nodo de destino.

```
export VMNAME=NAME-OF-THE-VIRTUAL-MACHINE
```

c Cree la variable de entorno `VMIP` cuyo valor sea la dirección IP de la máquina virtual del nodo de destino.

```
export VMIP=$(kubectl -n $NAMESPACE get virtualmachine/$VMNAME -o
jsonpath='{.status.vmIp}')
```

10 Para utilizar SSH en el nodo del clúster mediante el pod de Jump Box, ejecute el siguiente comando.

```
kubectl exec -it jumpbox  /usr/bin/ssh vmware-system-user@$VMIP
```

**Importante** La creación del contenedor e instalación del software tarda aproximadamente 60 segundos. Si recibe un "mensaje de error al ejecutar el comando en el contenedor: container_linux.go:370: starting container process caused: exec: "/usr/bin/ssh": stat /usr/bin/ssh: no such file or directory", vuelva a intentar el comando pasados unos segundos.

11 Para confirmar la autenticidad del host, introduzca **`yes`**.

```
The authenticity of host '10.249.0.999 (10.249.0.999)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.249.0.999' (ECDSA) to the list of known hosts.
Welcome to Photon 3.0
```

12 Confirme que ha iniciado sesión en el nodo de destino como `vmware-system-user`.

Por ejemplo, el siguiente resultado indica que ha iniciado sesión en un nodo del plano de control como usuario del sistema.

```
vmware-system-user@tkg-cluster-1-control-plane-66tbr [ ~ ]$
```

13 Realice las operaciones deseadas en el nodo.

**Atención** Es posible que deba usar `sudo` o `sudo su` para realizar ciertas operaciones en el nodo, como reiniciar el kubelet.

14 Cuando termine, escriba **`exit`** para cerrar la sesión de SSH en pod de vSphere.

15 Para eliminar el pod, ejecute el comando `kubectl delete pod jumpbox`.

**Precaución** Por motivos de seguridad, considere la posibilidad de eliminar el pod de Jumpbox después de haber realizado su trabajo. Si es necesario, puede volver a crearla más adelante.

# Conectarse mediante SSH a nodos de clúster de Servicio TKG como usuario del sistema con una contraseña

Es posible conectarse mediante SSH a un nodo de clúster de carga de trabajo como `vmware-system-user` mediante una contraseña.

Puede conectarse a un nodo de clúster como usuario `vmware-system-user` con una contraseña. La contraseña se almacena como un secreto denominado `CLUSTER-NAME-ssh-password`. La contraseña está codificada en Base64 en `.data.ssh-passwordkey`. Puede proporcionar la contraseña a través de una sesión de SSH. Consulte Obtener secretos de clúster de TKG mediante Kubectl.

### Requisitos previos

Para enrutar las conexiones SSH en la red de cargas de trabajo adecuada, implemente una máquina virtual de host de salto de Linux en el entorno de vSphere en el que está habilitada **Administración de cargas de trabajo**. Consulte Crear una máquina virtual de host de salto de Linux.

**Nota** La implementación de una máquina virtual de host de salto es un requisito estricto si utiliza redes de vDS y desea conectarse a nodos del clúster mediante SSH. También puede utilizar este enfoque con las redes NSX si prefiere utilizar una contraseña en lugar de una clave privada para conectarse a través de SSH.

### Procedimiento

1 Obtenga la dirección IP de la máquina virtual del host de salto, el nombre de usuario y la contraseña.

Consulte Crear una máquina virtual de host de salto de Linux.

2 Conéctese al Supervisor.

Consulte Conectarse a Supervisor como usuario de vCenter Single Sign-On con kubectl.

3 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
kubectl config use-context VSPHERE-NAMESPACE
```

4 Obtenga la dirección IP del nodo del clúster de destino.

   Enumere los nodos.

   ```
   kubectl get virtualmachines
   ```

   Describa los nodos para obtener la dirección IP del nodo de destino.

   ```
   kubectl describe virtualmachines
   ```

5 Observe el secreto de `TKG-CLUSTER-NAME-ssh-password`.

   ```
   kubectl get secrets
   ```

6 Obtenga el valor de ssh-passwordkey para el clúster de destino.

   ```
   kubectl get secrets TKG-CLUSTER-NAME-ssh-password -o yaml
   ```

   Por ejemplo, se devuelve ssh-passwordkey.

   ```
   apiVersion: v1
   data:
     ssh-passwordkey: RU1pQllLTC9TRjVFV0RBcCtmd1zwOTROeURYSWNGeXNReXJhaXRBUllYaz0=
   ```

7 Descodifique ssh-passwordkey.

   El secreto está codificado en Base64. Para descodificarlo: en Linux, utilice `base64 --decode` (o `base64 -d`); en MacOS, utilice `base64 --Decode` (o `base64 -D`); en Windows, utilice una herramienta en línea.

   ```
   echo <ssh-passwordkey> | base64 --decode
   ```

8 Conéctese mediante SSH al nodo del clúster de destino como `vmware-system-user`.

   ```
   ssh vmware-system-user@TKG-CLUSTER-NODE-IP-ADDRESS
   ```

9 Inicie sesión con la contraseña que descodificó.

## Crear una máquina virtual de host de salto de Linux

Para utilizar SSH en los nodos de clúster de carga de trabajo con una contraseña, cree una máquina virtual de Jump Box que se conecte a la red de cargas de trabajo y la red de administración o front-end para la tunelización de SSH.

### Crear una máquina virtual de host de salto de Linux

Siga estos pasos para crear una máquina virtual de Jump Box de Linux. Existen varias formas de hacerlo. Esta es una de ellas. Las instrucciones utilizan Photon OS, el cual se puede descargar aquí: https://github.com/vmware/photon/wiki/Downloading-Photon-OS.

**Nota** Este método de creación de un host de salto es para entornos de redes de vDS. Si utiliza NSX, cree un host de salto con pod de vSphere. Consulte Conectarse mediante SSH a nodos de clúster de Servicio TKG como usuario del sistema con una clave privada.

1 Inicie sesión en vCenter Server con vSphere Client.

2 Cree una máquina virtual nueva.

3 Seleccione el sistema operativo invitado Linux, que en este ejemplo es VMware Photon OS (64 bits).

4 Instale el sistema operativo. Para ello, descargue el archivo ISO, asócielo a la máquina virtual e inícielo.

5 Configure la máquina virtual con una dirección IP en Red de carga de trabajo > Red de espacio de nombres.

   **Nota** Es posible crear un conflicto de IP si el ámbito del rango de Red de carga de trabajo consume todo el espacio de red del grupo de puertos que utiliza.

6 Agregue una segunda NIC virtual a la máquina virtual y asígnela a la red de administración de front-end.

7 Complete la configuración del sistema operativo y encienda la máquina virtual después de reiniciar.

8 Inicie sesión en la consola de vSphere de la máquina virtual como usuario raíz.

9 Cree una interfaz de red para la nueva NIC y asígnele una dirección IP en la red de front-end.

```
ifconfig eth1 IP-ADDRESS netmask NETMASK up
```

   **Nota** Este método no es persistente durante los reinicios.

10 Compruebe que puede hacer ping en la puerta de enlace y el servidor DNS a través de esa interfaz.

11 En la consola de vSphere de la máquina virtual, configure un usuario de SSH con certificados. Para comprobar que funciona, cree un shell anidado.

12 Para verificar que funciona, ejecute SSH en Jump Box desde la red de front-end como el usuario de SSH.

13 Instale sshpass en la máquina virtual (de modo que pueda iniciar sesión a través de SSH con una contraseña). Para Photon OS, el comando es el siguiente:

```
tdnf install -y sshpass
```

14 Agregue la clave pública del cliente al archivo `~/.ssh/authorized_keys` y reinicie el proceso de `sshd` para que SSH pueda funcionar sin contraseña.

- Obtenga la clave pública, por ejemplo: `cat ~/.ssh/id_rsa.pub`.
- Acceda a la máquina virtual del host de acceso directo.
- Cree el directorio SSH (si no existe): `mkdir -p ~/.ssh`.
- Anexe la clave pública al archivo `authorized_keys`: `echo ssh-rsa AAAA.... >> ~/.ssh/authorized_keys`. Reemplace `ssh-rsa AAAA....` por la cadena de clave pública completa que se obtuvo desde el comando `cat ~/.ssh/id_rsa.pub`.
- Asegúrese de que el directorio `~/.ssh` y el archivo `authorized_keys` tengan establecidos los permisos adecuados, por ejemplo: `chmod -R go= ~/.ssh`.

# Crear un grupo y una función dedicados para los operadores de plataforma

Una forma opcional de asignar permisos a los operadores de vSphere IaaS control plane, incluidas las personas responsables de operar clústeres de Supervisor y Servicio TKG, es crear una función y un grupo de usuarios de vSphere dedicados para los operadores.

## Acerca de la función y del grupo de operadores de plataforma

Como práctica recomendada de seguridad, evite asignar la función de administrador de vSphere a los operadores de plataforma, ya que concede más privilegios de los necesarios para operar clústeres de Servicio TKG. Según el principio de privilegio mínimo, puede crear un grupo de usuarios dedicado, una cuenta de servicio (usuario) y una función personalizada para que utilicen los operadores de plataforma, y otorgar permisos de función personalizada al grupo de usuarios para objetos de vSphere.

**Nota** Debe iniciar sesión en vCenter Server como administrador de vSphere para realizar todas las tareas de este tema.

**Nota** Los nombres de grupo y funciones de vSphere son cadenas definidas por el usuario. Los nombres que se brindan aquí son ejemplos que puede adoptar, ajustar o cambiar según sus necesidades de seguridad y negocio.

**Advertencia** Debe evaluar los permisos de función de ejemplo brindados aquí en el contexto de los requisitos empresariales y de seguridad, probar la función para garantizar la conformidad y realizar los ajustes correspondientes. No todos los permisos que se utilizan aquí pueden ser adecuados para sus necesidades y es posible que necesite permisos adicionales. Consulte la documentación de Seguridad de vSphere para obtener una lista completa de los permisos y las consideraciones de seguridad.

## Parte 1: crear un usuario y un grupo de operadores de plataforma

En vCenter, o un sistema de AD/LDAP integrado con vCenter, cree el grupo de operadores de plataforma y una cuenta de usuario inicial.

1 Inicie sesión en vCenter Server como administrador mediante vSphere Client.

2 Vaya a **Administración > Single Sign On > Usuarios y grupos**.

3 Seleccione la pestaña **Grupos**.

4 Haga clic en **Agregar** y cree un nuevo grupo:

- Nombre: *platform-operators-group*
- Descripción: *Cuenta de grupo para Kubernetes Operators de clústeres de servicio TKG y Supervisor*
- Haga clic en **Agregar**

5 Seleccione la pestaña **Usuarios**.

6 Haga clic en **Agregar** y cree un nuevo usuario para fines de prueba.

- Nombre: *platform-operator-00*
- Contraseña: introduzca una contraseña segura que cumpla los requisitos
- Haga clic en **Agregar**

7 Seleccione la pestaña **Grupos**.

8 Agregue el nuevo usuario al grupo.

- Seleccione el grupo *platform-operators-group*.
- Haga clic en **Agregar miembros**.
- Seleccione **vsphere.local**.
- Busque el nombre de usuario *platform-operator-00*.
- Seleccione el usuario y haga clic en **Agregar**.
- Haga clic en **Guardar**.

## Parte 2: agregar el grupo de operadores de plataforma al grupo de usuarios del proveedor de servicios

Agregue el grupo de operadores de plataforma al grupo de usuarios del proveedor de servicios. Al hacerlo, los miembros del grupo de operadores de plataforma podrán ver espacios de nombres de vSphere en la pantalla de vCenter **Inventario > Hosts y clústeres**.

1 Vaya a **Administración > Single Sign On > Usuarios y grupos**.

2 Seleccione la pestaña **Grupos**.

3 Busque el grupo **ServiceProviderUsers**.

4 Edite el grupo **ServiceProviderUsers** y agregue **platform-operators-group** como miembro.

5 Haga clic en **Guardar**.

## Parte 3: crear la función de operadores de plataforma

Cree una función de vCenter SSO personalizada para los operadores de plataforma.

**Nota** La función incluye todos los permisos necesarios para aprovisionar y operar clústeres de Supervisor y de Servicio TKG, incluida la administración de bibliotecas de contenido. Es posible que deba ajustar los permisos asignados a esta función según sus requisitos empresariales y de seguridad. Además, debe probar la función para asegurarse de que cumple con sus requisitos.

1 Con vSphere Client, vaya a **Administración > Control de acceso > Funciones**.

2 Seleccione **Nuevo** y cree un nuevo nombre de función **`platform-operators-role`**.

3 Defina los siguientes privilegios para esta función.

4 Cuando termine, haga clic en **Guardar**.

```
- Alarms
  - Acknowledge alarm &
  - Create alarm &
  - Disable alarm action on entity &
  - Modify alarm &
  - Remove alarm &
  - Set alarm status &
- Certificate Authority
  - Create/Delete (below Admins priv) &
- Certificate Management
  - Create/Delete (below Admins priv) &
- Cns
  - Searchable * &
- Compute Policy
  - Create and Delete Compute Policy &
- Content Library
  - Add library item &
  - Check in a template &
  - Check out a template &
  - Create local library &
  - Create subscribed library &
  - Delete library item &
  - Delete local library &
  - Delete subscribed library &
  - Download files &
  - Evict library item &
  - Evict subscribed library &
  - Import storage &
  - Probe subscription information &
  - Read storage &
  - Sync library item &
  - Sync subscribed library &
  - Type introspection &
```

```
    - Update configuration settings &
    - Update library &
    - Update library item &
    - Update local library &
    - Update subscribed library &
    - View configuration settings &
  - Datastore
    - Allocate space * & $
    - Browse datastore * &
    - Configure datastore &
    - Low level file operations * &
    - Remove file &
    - Rename datastore &
    - Update virtual machine files &
    - Update virtual machine metadata &
  - Extension
    - Register extension &
    - Unregister extension &
    - Update extension &
  - Folder
    - Create folder &
    - Delete folder &
    - Move folder &
    - Rename folder &
  - Global
    - Cancel task &
    - Disable methods * &
    - Enable methods * &
    - Global tag &
    - Health &
    - Licenses * &
    - Log event &
    - Manage custom attributes &
    - Service managers &
    - Set custom attribute &
    - System tag &
  - Host
    - Configuration
      - Network configuration $
  - Host profile
    - View &
  - Hybrid Linked Mode
    - Manage &
  - Namespaces
    - Modify cluster-wide configuration
    - Modify cluster-wide namespace self-service configuration
    - Modify namespace configuration
  - Network
    - Assign network * & $
  - Resource
    - Apply recommendation &
    - Assign vApp to resource pool * &
    - Assign virtual machine to resource pool &
    - Create resource pool &
    - Modify resource pool &
```

```
  - Move resource pool &
  - Query vMotion &
  - Remove resource pool &
  - Rename resource pool &
- Scheduled task
  - Create tasks &
  - Modify task &
  - Remove task &
  - Run task &
- Sessions
  - Message * &
  - Validate session * &
- VM storage policies
  - View VM storage policies *
- Storage views
  - View &
- Supervisor Services
  - Manage Supervisor Services
- Trusted Infrastructure administrator
  - Manage Trusted Infrastructure Hosts &
- vApp
  - Add virtual machine &
  - Assign resource pool &
  - Assign vApp &
  - Clone &
  - Create &
  - Delete &
  - Export &
  - Import * $
  - Move &
  - Power off &
  - Power on &
  - Rename &
  - Suspend &
  - Unregister &
  - View OVF environment &
  - vApp application configuration &
  - vApp instance configuration &
  - vApp managedBy configuration &
  - vApp resource configuration &
- Virtual machine
  - Change Configuration
    - Acquire disk lease &
    - Add existing disk * & $
    - Add new disk * &
    - Add or remove device * &
    - Advanced configuration * & $
    - Change CPU count * &
    - Change Memory * &
    - Change Settings * &
    - Change Swapfile placement &
    - Change Resource &
    - Configure Host USB device &
    - Configure Raw device * &
    - Configure managedBy &
```

```
    - Display connection settings &
    - Extend virtual disk * &
    - Modify device settings * &
    - Query Fault Tolerance compatibility &
    - Query unowned files &
    - Reload from path &
    - Remove disk * &
    - Rename &
    - Reset guest information &
    - Set annotation &
    - Toggle disk change tracking * &
    - Upgrade virtual machine compatibility &
  - Edit Inventory
    - Create from existing * &
    - Create new &
    - Move &
    - Remove * &
    - Register &
    - Unregister &
  - Guest operations
    - Guest operation alias modification &
    - Guest operation alias query &
    - Guest operation modifications &
    - Guest operation program execution &
    - Guest operation queries &
  - Interaction
    - Answer question &
    - Backup operation on virtual machine &
    - Configure CD media &
    - Configure floppy media &
    - Connect devices &
    - Console interaction &
    - Create screenshot &
    - Defragment all disks &
    - Drag and drop &
    - Guest operating system management by VIX API &
    - Inject USB HID scan codes &
    - Install VMware Tools &
    - Pause or Unpause &
    - Power off * &
    - Power on * &
    - Reset &
    - Suspend &
  - Provisioning
    - Allow disk access &
    - Allow file access &
    - Allow read-only disk access * &
    - Allow virtual machine download * &
    - Allow virtual machine files upload &
    - Clone template &
    - Clone virtual machine &
    - Create template from virtual machine &
    - Customize guest &
    - Deploy template * &
    - Mark as template &
```

```
    - Mark as virtual machine &
    - Modify customization specification &
    - Promote disks &
    - Read customization specifications &
  - Service configuration
    - Allow notifications &
    - Allow polling of global event notifications &
    - Manage service configurations &
    - Modify service configuration &
    - Query service configurations &
    - Read service configuration &
  - Snapshot management
    - Create snapshot * &
    - Remove snapshot * &
    - Rename snapshot &
    - Revert to snapshot &
  - vSphere Replication
    - Configure replication &
    - Manage replication &
    - Monitor replication &
- Virtual Machine Classes
  - Manage Virtual Machine Classes
- vSan
  - Cluster &
    - ShallowRekey &
- vService
  - Create dependency &
  - Destroy dependency &
  - Reconfigure dependency configuration &
  - Update dependency &
- vSphere Tagging
  - Assign or Unassign vSphere Tag &
  - Assign or Unassign vSphere Tag on Object &
  - Create vSphere Tag &
  - Create vSphere Tag Category &
  - Delete vSphere Tag &
  - Delete vSphere Tag Category &
  - Edit vSphere Tag &
  - Edit vSphere Tag Category &
  - Modify UsedBy Field For Category &
  - Modify UsedBy Field For Tag &
```

## Parte 4: asignar permisos de objeto de vCenter a la función y el grupo de operadores de plataforma

Asigne permisos al grupo de operadores de plataforma para los objetos de vCenter que utilizan clústeres de Supervisory de Servicio TKG.

1 En vCenter, seleccione la vista **Inventario**.

2 Para cada uno de los objetos de vCenter enumerados a continuación, haga clic con el botón secundario en el objeto y seleccione **Agregar permiso**.

3 Para Usuario/Grupo, seleccione el grupo **platform-operators-group**.

4 Para Función, seleccione la función **platform-operators-group-role**.

5 Para algunos objetos, deberá seleccionar **Propagar a objetos secundarios**, como se indica.

- **Hosts y clústeres**
  - El objeto raíz de vCenter Server.
  - El centro de datos y todas las carpetas Host y Clúster desde el objeto Centro de datos hasta el clúster que administra la implementación de TKG.
  - El clúster de vCenter de destino donde se habilitó el Supervisor, con la opción **Propagar a objetos secundarios** habilitada (para los hosts ESXi, etc.).
  - Los grupos de recursos de destino con la opción **Propagar a objetos secundarios** habilitada.
- **Máquinas virtuales y plantillas**
  - El objeto Centro de datos de nivel superior, con la opción **Propagar a objetos secundarios** habilitada.
  - Como alternativa, para afinar la granularidad, las carpetas Máquina virtual y Plantilla de destino con la opción **Propagar a objetos secundarios** habilitada.
- **Almacenamiento**
  - El objeto Centro de datos de nivel superior, con la opción **Propagar a objetos secundarios** habilitada.
  - De forma alternativa, el objeto Almacén de datos compartido (por ejemplo, vsanDatastore) sin la opción **Propagar a objetos secundarios** habilitada, o almacenes de datos individuales y todas las carpetas de almacenamiento, desde el objeto Centro de datos hasta los almacenes de datos que se utilizarán para las implementaciones de TKG, con la opción **Propagar a objetos secundarios** habilitada.
- **Redes**
  - El objeto Centro de datos de nivel superior, con la opción **Propagar a objetos secundarios** habilitada.
  - Como alternativa, las redes individuales, los conmutadores distribuidos y los grupos de puertos distribuidos a los que se asignarán clústeres.

## Parte 5: asociar la función y el grupo de operadores de plataforma

Asigne permisos a la función y al grupo de operadores de plataforma.

1 Con vSphere Client, vaya a **Administración > Control de acceso > Permiso global > Agregar permiso**.

2 Agregue la función de operadores de plataforma al grupo de operadores de plataforma.

   - Haga clic en **Agregar**.
   - Para Dominio, seleccione **vsphere.local**.

- Para Usuario/Grupo, introduzca el grupo **platform-operators-group**.
- Para la Función, seleccione la función **platform-operators-role**.
- Seleccione la casilla de verificación **Propagar a objetos secundarios**.
- Haga clic en **ACEPTAR** para actualizar el grupo con los permisos de la función.

3 Agregue la función Administrador de Kubernetes de vSphere al grupo de operadores de plataforma.

- Haga clic en **Agregar**.
- Para Dominio, seleccione **vsphere.local**.
- Para Usuario/Grupo, introduzca el grupo **platform-operators-group**.
- Para Función, seleccione la función **Administrador de Kubernetes de vSphere**.
- Seleccione la casilla de verificación **Propagar a objetos secundarios**.
- Haga clic en **ACEPTAR** para actualizar el grupo con los permisos de la función.

## Parte 6: validar la función y el grupo de operadores de plataforma

Pruebe la función y el grupo de operadores nuevos de plataforma. Debe asegurarse de que el grupo y la función cumplan con los requisitos empresariales y de seguridad, y realizar los ajustes correspondientes.

1 Con vSphere Client, inicie sesión en vCenter Server con la cuenta de usuario de **platform-operator-00**.

2 Compruebe que tiene acceso de lectura a los objetos de vSphere, incluidos hosts y clústeres, máquinas virtuales y plantillas, almacenamiento y redes.

3 Compruebe que puede crear y configurar una biblioteca de contenido. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

4 Compruebe que puede crear y configurar una instancia de espacio de nombres de vSphere, incluida la adición de usuarios, la asociación de la biblioteca de contenido y la asignación de directivas de almacenamiento. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

5 Compruebe que puede iniciar sesión en Supervisor con Herramientas de la CLI de Kubernetes para vSphere. Consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

6 Compruebe que puede aprovisionar un clúster de TKG en Supervisor con Kubectl. Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

7 Compruebe que puede iniciar sesión en el clúster de TKG en Supervisor con Kubectl. Consulte Conectarse a un clúster de Servicio TKG como usuario de vCenter Single Sign-On con Kubectl.

8 Compruebe que puede implementar cargas de trabajo en el clúster de TKG en Supervisor. Consulte Capítulo 12 Implementar cargas de trabajo en clústeres de Servicio TKG.

9 Compruebe que puede realizar diferentes tareas operativas para el clúster de TKG en Supervisor. Consulte Capítulo 8 Operar clústeres del servicio de TKG.

10 Con vSphere Client, compruebe que puede ver el clúster de TKG aprovisionado en Supervisor en espacio de nombres de vSphere.

11 Con vSphere Client, compruebe que puede supervisar Supervisor y objetos de clúster de TKG.

12 Realice una prueba negativa para asegurarse de que no puede realizar ciertas tareas reservadas para administradores de vSphere. Por ejemplo, no debería poder crear una nueva directiva de almacenamiento de vSphere o redes de vSphere. Tampoco debe poder deshabilitar la administración de cargas de trabajo.

13 Ajuste los permisos de función según corresponda para cumplir con sus requisitos empresariales y de seguridad .

# Administrar las versiones de Kubernetes para clústeres de Servicio TKG

5

versiones de Tanzu Kubernetes (TKr) proporciona la distribución de software de Kubernetes para los clústeres de Servicio TKG. VMware distribuye las Tkr como plantillas de máquina virtual, las cuales se sincronizan con la plataforma mediante una biblioteca de contenido de vCenter.

Lea los siguientes temas a continuación:



## Uso de versiones de Kubernetes con clústeres de Servicio TKG

Una versión de Tanzu Kubernetes (TKr) proporciona la distribución de software de Kubernetes firmada y compatible con VMware para su uso con clústeres de Servicio TKG. El formato TKr se actualizó para que vSphere 8 admita paquetes y varios sistemas operativos.

### Notas de la versión de TKr

Consulte las Notas de la versión de las distintas versiones de Tanzu Kubernetes para obtener la lista completa de TKr disponibles, las novedades de cada versión, los problemas conocidos y la compatibilidad de TKr.

## Distribución y uso de TKr

VMware distribuye versiones de Tanzu Kubernetes a través de una red de entrega de contenido. Utilice una biblioteca de contenido vSphere para asociar TKr con espacios de nombres de vSphere. Para automatizar el uso de TKr, utilice una Crear una biblioteca de contenido suscrita. Para entornos con restricciones de Internet, utilice una Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados).

Cada versión de Tanzu Kubernetes se distribuye como una plantilla de OVA. La controladora de TKr en Supervisor utiliza la plantilla de OVA a fin de crear las máquinas virtuales para los nodos del clúster de TKG. El tamaño de disco de la máquina virtual se establece mediante la plantilla de OVA de TKr. Los recursos de CPU y RAM se especifican mediante Uso de clases de máquinas virtuales con clústeres de Servicio TKG.

La biblioteca de contenido de TKr no está en el ámbito del espacio de nombres. Todas las instancias de espacios de nombres de vSphere utilizan la misma biblioteca de contenido de TKr para TKG en Supervisor. Al editar la biblioteca de contenido de TKr para una instancia de espacio de nombres de vSphere, se actualizará para el resto de las instancias.

## Cadena TKr NAME

Utilice el comando `kubectl get tkr` para enumerar las imágenes de TKr que están disponibles y pueden usarse en espacio de nombres de vSphere. Por ejemplo:

```
kubectl get tkr
NAME                                  VERSION                               READY
COMPATIBLE   CREATED
v1.16.14---vmware.1-tkg.1.ada4837     v1.16.14+vmware.1-tkg.1.ada4837       False
False        19d
v1.17.17---vmware.1-tkg.1.d44d45a     v1.17.17+vmware.1-tkg.1.d44d45a       False
False        19d
v1.18.19---vmware.1-tkg.1.17af790     v1.18.19+vmware.1-tkg.1.17af790       False
False        19d
v1.19.16---vmware.1-tkg.1.df910e2     v1.19.16+vmware.1-tkg.1.df910e2       False
False        19d
v1.20.12---vmware.1-tkg.1.b9a42f3     v1.20.12+vmware.1-tkg.1.b9a42f3       False
False        19d
v1.21.6---vmware.1-tkg.1.b3d708a      v1.21.6+vmware.1-tkg.1.b3d708a        True
True         19d
v1.22.9---vmware.1-tkg.1.cc71bc8      v1.22.9+vmware.1-tkg.1.cc71bc8        True
True         19d
v1.23.8---vmware.2-tkg.2-zshippable   v1.23.8+vmware.2-tkg.2-zshippable     True
True         19d
v1.23.8---vmware.3-tkg.1              v1.23.8+vmware.3-tkg.1                True
True         19d
```

La cadena TKr NAME se utiliza para aprovisionar clústeres de TKG en Supervisor. Si utiliza la API v1alpha3 del clúster de Tanzu Kubernetes, proporcione la cadena completa TKr NAME en el campo `tkr.reference.name`. Si utiliza la API de clúster v1beta1 , proporcione la cadena completa TKr NAME en el campo `topology.version`.

**Nota** No utilice la cadena VERSION al hacer referencia a la TKr en la especificación del clúster. El formato debe coincidir exactamente con la cadena TKr NAME.

El nombre de la TKr en la biblioteca de contenido debe ser la cadena TKr NAME completa. Si utiliza una biblioteca de contenido suscrita, la cadena TKr NAME se crea automáticamente. Si utiliza una biblioteca de contenido local, asegúrese de que el nombre que asigne a la TKr coincida con la cadena TKr NAME. Consulte Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados) para obtener más información.

## Compatibilidad de TKr con Servicio TKG

Las TKr se publican y se actualizan de forma independiente del Servicio TKG y del Supervisor.

Para aprovisionar un clúster de TKG, la TKr debe ser compatible con el Servicio TKG. No puede utilizar una TKr que no sea compatible con el Servicio TKG. Además, debe asegurarse de ejecutar una TKr compatible para la versión de destino a la que desea actualizar.

Puede comprobar la compatibilidad de la TKr mediante el comando `kubectl get tkr`. La columna COMPATIBLE devuelve un valor booleano. `True` significa que la TKr es compatible y `False` significa que la TKr no es compatible.

## Compatibilidad de TKR con vSphere

El formato TKr se ha actualizado para vSphere 8. Las TKr para vSphere 8 solo se pueden ejecutar en vSphere 8.x. Las TKr para vSphere 7.x son imágenes heredadas que funcionan con vSphere 7. Estas imágenes se pueden ejecutar en vSphere 8, pero solo para fines de actualización. Las imágenes de TKr heredadas se identifican mediante la etiqueta de anotación `legacy-tkr`.

Puede comprobar la compatibilidad de la TKr mediante los comandos `kubectl get tkr -o yaml` y `kubectl get tkr --show-labels`. Si la etiqueta de anotación `legacy-tkr` está presente, la TKr no admite las funciones de vSphere 8 y solo se debe utilizar para actualizar a vSphere 8 desde vSphere 7.

Por ejemplo, el siguiente comando muestra que la imagen especificada es un `legacy-tkr`.

```
kubectl get tkr v1.23.8---vmware.3-tkg.1 -o yaml
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesRelease
metadata:
  creationTimestamp: "2023-03-15T20:33:17Z"
  finalizers:
  - tanzukubernetesrelease.run.tanzu.vmware.com
  generation: 1
  labels:
    os-arch: amd64
```

```
    os-name: photon
    os-type: linux
    os-version: "3.0"
    run.tanzu.vmware.com/legacy-tkr: ""
    v1: ""
    v1.23: ""
    v1.23.8: ""
    v1.23.8---vmware: ""
    v1.23.8---vmware.3: ""
    v1.23.8---vmware.3-tkg: ""
    v1.23.8---vmware.3-tkg.1: ""
  name: v1.23.8---vmware.3-tkg.1
```

En el siguiente ejemplo se utiliza la marca `--show-labels` para comprobar la compatibilidad de la TKr. La etiqueta `legacy-tkr` está presente, por lo que la imagen solo se puede utilizar para crear un clúster de TKG heredado.

```
kubectl get tkr v1.23.8---vmware.3-tkg.1 --show-labels
NAME                       VERSION                  READY   COMPATIBLE   CREATED   LABELS
v1.23.8---vmware.3-tkg.1   v1.23.8+vmware.3-tkg.1   True    True         19d       os-
arch=amd64,os-name=photon,os-type=linux,os-version=3.0,run.tanzu.vmware.com/legacy-tkr=,
```

El siguiente ejemplo muestra que una TKr está diseñada específicamente para vSphere 8.x porque la etiqueta `legacy-tkr` está ausente en la lista de etiquetas.

```
kubectl get tkr v1.28.8---vmware.1-fips.1-tkg.2 --show-labels
NAME                              VERSION                         READY   COMPATIBLE
CREATED   LABELS
v1.28.8---vmware.1-fips.1-tkg.2   v1.28.8+vmware.1-fips.1-tkg.2   True    True
21d       os-arch=amd64,os-name=photon,os-type=linux,os-version=5.0,tkr.tanzu.vmware.com/
standard=,v1.28.8---vmware.1-fips.1-tkg.2=,v1.28.8---vmware.1-fips.1-tkg=,v1.28.8---vmware.1-
fips.1=,v1.28.8---vmware.1-fips=,v1.28.8---vmware.1=,v1.28.8---vmware=,v1.28.8=,v1.28=,v1=
```

Al ejecutar el comando equivalente con `-o yaml` también se muestra que la etiqueta `legacy-tkr` está ausente, lo que indica que la TKr está diseñada para vSphere 8.x.

```
 kubectl get tkr v1.28.8---vmware.1-fips.1-tkg.2 -o yaml
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesRelease
metadata:
  creationTimestamp: "2024-05-08T20:03:57Z"
  finalizers:
  - tanzukubernetesrelease.run.tanzu.vmware.com
  generation: 2
  labels:
    os-arch: amd64
    os-name: photon
    os-type: linux
    os-version: "5.0"
    tkr.tanzu.vmware.com/standard: ""
    v1: ""
    v1.28: ""
    v1.28.8: ""
```

```
    v1.28.8---vmware: ""
    v1.28.8---vmware.1: ""
    v1.28.8---vmware.1-fips: ""
    v1.28.8---vmware.1-fips.1: ""
    v1.28.8---vmware.1-fips.1-tkg: ""
    v1.28.8---vmware.1-fips.1-tkg.2: ""
  name: v1.28.8---vmware.1-fips.1-tkg.2
...
```

## Formato TKr Imagen de sistema operativo

El formato TKr Imagen de sistema operativo admite varias imágenes de sistema operativo para una sola TKr. Esto significa que hay una única versión de Tanzu Kubernetes para una versión específica de Kubernetes para todos los sistemas operativos compatibles, actualmente Photon OS y Ubuntu. El formato predeterminado de imagen de sistema operativo es PhotonOS.

De forma predeterminada, la edición Photon OS de la TKr designada se utiliza para los nodos de clúster de TKG. Si la TKr a la que se hace referencia admite el formato Imagen de SO y tiene una edición del sistema operativo Ubuntu disponible, utilice la anotación `run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu` para especificar la edición del sistema operativo Ubuntu de TKr. Por ejemplo, la siguiente Capítulo 7 Aprovisionar clústeres del servicio de TKG utiliza la edición Ubuntu de la TKr designada.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-ubuntu
  namespace: tkgs-cluster-ns
  annotations:
    run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
spec:
   topology:
     controlPlane:
       ...
       tkr:
         reference:
           name: v1.28.8---vmware.1-fips.1-tkg.2
```

El formato Imagen de SO admite implementaciones de clústeres heterogéneos. Por ejemplo, el siguiente manifiesto del clúster crea un clúster de Tanzu Kubernetes con el PhotonOS predeterminado para los nodos del plano de control y Ubuntu para los nodos de trabajo. La versión de TKr se menciona en la sección del plano de control y la anotación especifica Ubuntu para el grupo de nodos de trabajo designado.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-multiOS
  namespace: tkgs-cluster-ubuntu
  annotations:
    //Worker nodes annotation
```

```
    run.tanzu.vmware.com/resolve-os-image.np-1: os-name=ubuntu
spec:
  topology:
    controlPlane:
      tkr:
        reference:
          name: v1.28.8---vmware.1-fips.1-tkg.2
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkgs-storage-profile
    nodePools:
    - replicas: 3
      name: np-1
      vmClass: guaranteed-medium
      storageClass: tkgs-storage-profile
```

Cuando el sistema se actualiza a vSphere 8, las TKr existentes se convierten automáticamente al formato Imagen de sistema operativo TKr con referencia a una sola imagen del sistema operativo. Esto permite que las TKr heredadas sean compatibles para la actualización a una TKr no heredada.

Las TKr heredadas tienen dos ediciones: Photon y Ubuntu. Cuando la TKr heredada tiene una edición específica de Ubuntu, puede utilizar la cadena de versión completa (con "ubuntu") y omitir la etiqueta de anotación, u utilizar la cadena de versión corta (sin "ubuntu") e incluir la etiqueta de versión.

## Paquetes de TKr

Las imágenes de TKr compatibles con vSphere 8 se actualizan a un marco basado en paquetes para los componentes básicos, como la interfaz de almacenamiento de contenedor (Container Storage Interface, CSI) y la interfaz de red de contenedor (Container Network Interface, CNI). Si utiliza la API v1beta, el cambio o la actualización de estos componentes se realiza mediante definiciones de recursos personalizados.

Para ver los paquetes que componen una TKr, ejecute el siguiente comando:

```
kubectl get tkr TKR-NAME -o yaml
```

Por ejemplo:

```
kubectl get tkr v1.28.8---vmware.1-fips.1-tkg.2 -o yaml
```

El comando devuelve todos los paquetes de TKr. Por ejemplo:

```
spec:
  bootstrapPackages:
  - name: antrea.tanzu.vmware.com.1.13.3+vmware.3-tkg.2-vmware
  - name: vsphere-pv-csi.tanzu.vmware.com.3.1.0+vmware.1-tkg.6-vmware
  - name: vsphere-cpi.tanzu.vmware.com.1.28.0+vmware.1-tkg.1-vmware
  - name: kapp-controller.tanzu.vmware.com.0.48.2+vmware.1-tkg.1-vmware
  - name: guest-cluster-auth-service.tanzu.vmware.com.1.3.3+tkg.1-vmware
  - name: metrics-server.tanzu.vmware.com.0.6.2+vmware.3-tkg.5-vmware
```

```
  - name: secretgen-controller.tanzu.vmware.com.0.15.0+vmware.1-tkg.1-vmware
  - name: pinniped.tanzu.vmware.com.0.25.0+vmware.2-tkg.1-vmware
  - name: capabilities.tanzu.vmware.com.0.32.1+vmware.1
  - name: gateway-api.tanzu.vmware.com.1.0.0+vmware.1-tkg.1-vmware
  - name: calico.tanzu.vmware.com.3.26.3+vmware.1-tkg.1-vmware
```

Consulte Ejemplo de v1beta1: clúster con CNI de Calico para ver un ejemplo de caso práctico.

## Migrar tipos de sistema operativo de TKr

No se admiten actualizaciones de clústeres in situ entre sistemas operativos de TKr. Esto significa, por ejemplo, que no se puede actualizar un clúster de TKG mediante TKr v1.27.11 de Photon a TKr v1.28.8 de Ubuntu.

Si desea cambiar el tipo de sistema operativo de TKr que utiliza un clúster de TKG, tenga en cuenta el siguiente procedimiento. En este ejemplo, el clúster de origen utiliza TKr de Photon y el destino es TKr de Ubuntu.

- Con Velero, haga una copia de seguridad de las cargas de trabajo del clúster de TKG basado en Photon.

  Consulte Capítulo 20 Copia de seguridad y restauración de cargas de trabajo y clústeres de servicio TKG.

- Aprovisione un nuevo clúster de TKG mediante la TKr de Ubuntu.

  Consulte Capítulo 7 Aprovisionar clústeres del servicio de TKG.

- Con Velero, restaure las cargas de trabajo del clúster de TKG en el clúster de Ubuntu.

  Consulte Capítulo 20 Copia de seguridad y restauración de cargas de trabajo y clústeres de servicio TKG.

## Fortalecimiento de TKr

Hay disponibles guías de implementación técnica de seguridad (STIG) para los componentes del sistema, incluido el Supervisor y las TKr. Consulte Fortalecimiento de STIG de Tanzu para obtener más información.

## Crear su propia TKr

A partir de TKr v1.25.7 para vSphere 8.x, puede crear imágenes de máquina TKr personalizadas para clústeres de TKG en vSphere 8. Una imagen de máquina personalizada empaqueta un sistema operativo y una versión compatibles, una versión de Kubernetes basada en una TKr publicada y cualquier personalización que realice.

Para crear imágenes de máquina personalizadas para los nodos de clúster de TKG, utilice Tanzu Kubernetes Grid Image Builder de vSphere. Consulte la documentación para obtener más información sobre la creación de imágenes personalizadas, las versiones de TKr compatibles y las personalizaciones admitidas.

# Permisos de función necesarios para administrar bibliotecas de contenido

Se requieren varios permisos de función de vSphere para administrar bibliotecas de contenido. Si no utiliza la función de administrador de vSphere, consulte la lista de permisos según sea necesario.

## Permisos necesarios para administrar bibliotecas de contenido

La administración de bibliotecas de contenido requiere los siguientes permisos de función de vSphere.

Estos permisos se incluyen en la función de administrador de vSphere.

Para crear una función dedicada que incluya estos permisos, consulte Crear un grupo y una función dedicados para los operadores de plataforma.

```
- Content Library
  - Add library item &
  - Check in a template &
  - Check out a template &
  - Create local library &
  - Create subscribed library &
  - Delete library item &
  - Delete local library &
  - Delete subscribed library &
  - Download files &
  - Evict library item &
  - Evict subscribed library &
  - Import storage &
  - Probe subscription information &
  - Read storage &
  - Sync library item &
  - Sync subscribed library &
  - Type introspection &
  - Update configuration settings &
  - Update library &
  - Update library item &
  - Update local library &
  - Update subscribed library &
  - View configuration settings &
```

# Crear una biblioteca de contenido suscrita

Para aprovisionar clústeres de TKG en Supervisor, puede crear una biblioteca de contenido suscrita y sincronizar versiones de Tanzu Kubernetes. Una biblioteca de contenido suscrita permite automatizar la distribución de TKR y mantenerse al día con las versiones más recientes.

Para TKG en Supervisor, VMware publica versiones de Tanzu Kubernetes en una red de entrega de contenido. Puede crear una biblioteca de contenido que se suscriba a estas publicaciones de imagen. Puede elegir el modo de sincronización: inmediato o a petición.

**Advertencia** La sincronización inmediata de versiones de Tanzu Kubernetes en la red de entrega de contenido público puede demandar mucho tiempo y espacio de disco.

### Requisitos previos

La funcionalidad de la biblioteca de contenido es una función de vCenter Server de la que depende TKG en Supervisor. Para obtener más información, consulte "Usar bibliotecas de contenido" en la documentación de Administrar máquinas virtuales de vSphere.

### Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Menú > Bibliotecas de contenido**.

3 Haga clic en **Crear**.

   Se abrirá el asistente **Nueva biblioteca de contenido**.

4 Especifique **Nombre y ubicación** de la biblioteca de contenido y haga clic en **Siguiente** cuando haya terminado.

| Campo | Descripción |
| --- | --- |
| **Nombre** | Introduzca un nombre descriptivo, como `TKR-sub`. |
| **Notas** | Incluya una descripción, como `Biblioteca de suscripción para las TKR de TKG 2`. |
| **vCenter Server** | Seleccione la instancia de vCenter Server en la que está habilitada la **administración de cargas de trabajo**. |

5 Configure la suscripción de la biblioteca de contenido en la página **Configurar biblioteca de contenido** y haga clic en **Siguiente** cuando haya terminado.

   a Seleccione la opción **Biblioteca de contenido suscrita**.

   b Introduzca la dirección **URL de suscripción** del publicador.

   ```
   https://wp-content.vmware.com/v2/latest/lib.json
   ```

   c Para la opción **Descargar contenido**, seleccione una de las siguientes opciones:

| Opción | Descripción |
| --- | --- |
| **Inmediatamente** | El proceso de suscripción sincroniza tanto los metadatos como las imágenes de la biblioteca. Si se eliminan elementos de la biblioteca publicada, su contenido se conserva en el almacenamiento de la biblioteca suscrita, por lo que deberá eliminarlo de forma manual. |
| **Cuando sea necesario** | El proceso de suscripción sincroniza solo los metadatos de la biblioteca. Tanzu Kubernetes Grid Service descarga las imágenes cuando se publican. Cuando ya no necesita el elemento, puede eliminar el contenido del elemento para liberar espacio de almacenamiento. Para ahorrar almacenamiento, se recomienda esta opción. |

6 Cuando se le pida, acepte la huella digital del certificado SSL.

   El certificado SSL se almacena en su sistema hasta que la biblioteca de contenido suscrita se elimine del inventario.

7 Configure la directiva de seguridad de OVF en la página **Aplicar directiva de seguridad** y haga clic en **Siguiente** cuando haya terminado.

   a Seleccione **Aplicar directiva de seguridad**.

   b Seleccione la **directiva predeterminada de OVF**.

   Al seleccionar esta opción, el sistema comprueba el certificado de firma de OVF durante el proceso de sincronización. Una plantilla de OVF que no supera la validación del certificado se marca con la etiqueta de **error en la verificación**. Los metadatos de la plantilla se conservan, pero los archivos OVF no se pueden sincronizar.

   **Nota** Actualmente, la **directiva predeterminada de OVF** es la única directiva de seguridad compatible.

8 En la página **Agregar almacenamiento**, seleccione un almacén de datos como ubicación de almacenamiento para el contenido de la biblioteca de contenido y haga clic en **Siguiente**.

9 En la página **Listo para completar**, revise los detalles y haga clic en **Finalizar**.

10 En la página **Bibliotecas de contenido**, seleccione la nueva biblioteca de contenido que creó.

11 Confirme o complete la sincronización del contenido de la biblioteca.

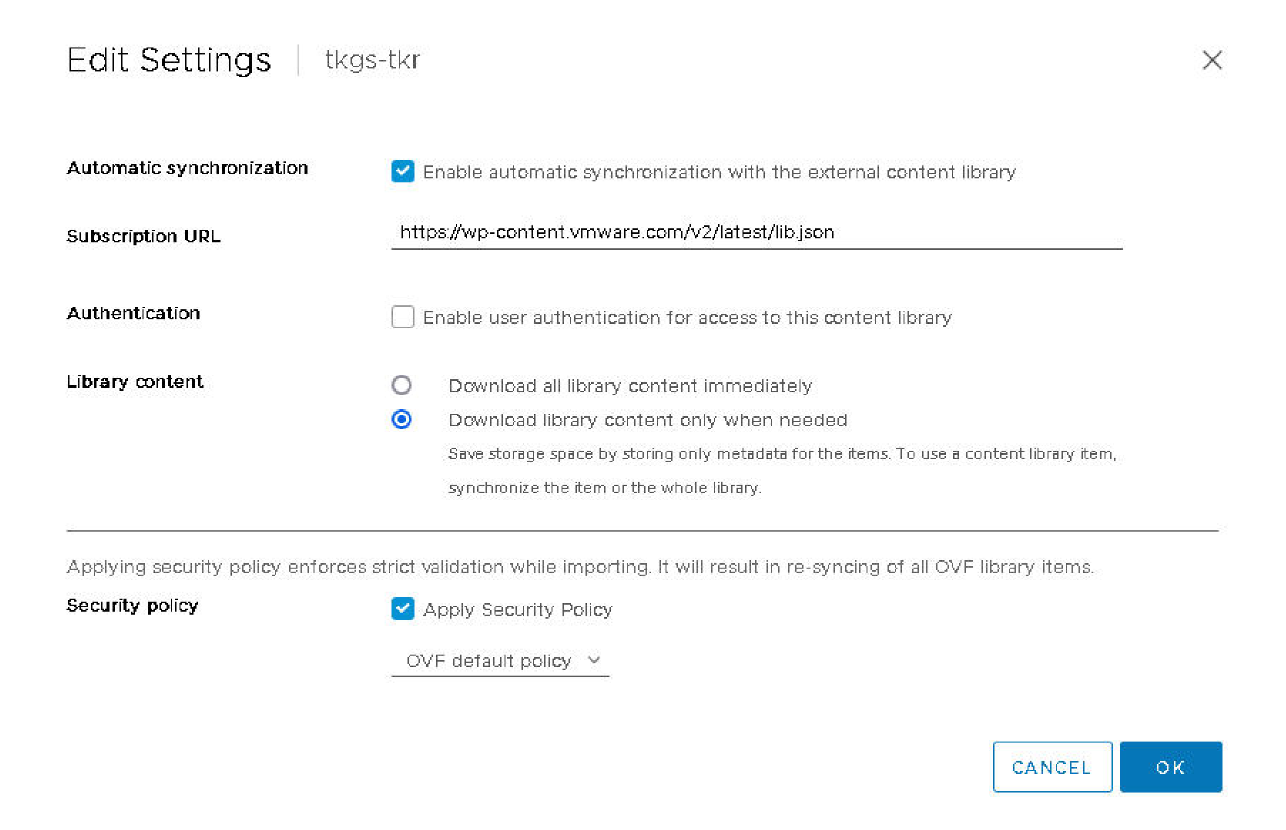
| Opción de sincronización | Descripción |
| --- | --- |
| **Inmediatamente** | Si eligió descargar todo el contenido inmediatamente, confirme que la biblioteca está sincronizada.<br>Para ver el contenido de la biblioteca sincronizada, seleccione **Plantillas > Plantillas de OVF y OVA**. |
| **Cuando sea necesario** | Si optó por sincronizar la biblioteca a petición, tiene dos opciones:<br>■ Utilizar **Acciones** > **Sincronizar** toda la biblioteca<br>■ Hacer clic con el botón secundario en un elemento y seleccionar **Sincronizar** para sincronizar solo eso<br>Para ver el contenido de la biblioteca sincronizada, seleccione **Plantillas > Plantillas de OVF y OVA**. |

12 Si eligió la opción **Cuando sea necesario**, descargue las plantillas de OVF que desee utilizar.

Si eligió la opción **Cuando sea necesario**, verá que los archivos de imagen no se almacenan localmente, sino que solo se almacenan los metadatos. Para descargar los archivos de plantilla, seleccione el elemento, haga clic con el botón secundario y seleccione **Sincronizar elemento**.

13 Para actualizar la configuración de la biblioteca de contenido suscrita, seleccione **Acciones > Editar configuración**.

| Configuración | Valor |
| --- | --- |
| URL de suscripción | `https://wp-content.vmware.com/v2/latest/lib.json` |
| Autenticación | No habilitado |
| Biblioteca de contenido | Descargar cuando sea necesario |
| directiva de seguridad | Directiva predeterminada de OVF |




#### Pasos siguientes

La biblioteca de contenido de la versión de Tanzu Kubernetes debe asociarse con cada espacio de nombres de vSphere donde se aprovisionen clústeres de TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

## Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados)

Para aprovisionar clústeres de TKG en Supervisor, puede crear una biblioteca de contenido local e importar versiones de Tanzu Kubernetes. El caso práctico típico de una biblioteca de contenido local es el de entornos con restricciones de Internet (aislados).

La creación de una biblioteca de contenido local implica configurar la biblioteca, descargar los archivos OVA e importarlos a la biblioteca de contenido local.

#### Requisitos previos

La funcionalidad de la biblioteca de contenido es una función de vCenter Server de la que depende TKG en Supervisor. Para obtener más información, consulte Usar bibliotecas de contenido.

#### Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Haga clic en **Menú**.

3 Haga clic en **Biblioteca de contenido**.

4 Haga clic en **Crear**.

El sistema muestra el asistente **Nueva biblioteca de contenido**.

5 Especifique **Nombre y ubicación** de la biblioteca de contenido y haga clic en **Siguiente** cuando haya terminado.

| Campo | Descripción |
|---|---|
| **Nombre** | Introduzca un nombre descriptivo, como `TKr-local`. |
| **Notas** | Incluya una descripción, como `Biblioteca local para las TKr de TKG` |
| **vCenter Server** | Seleccione la instancia de vCenter Server en la que vSphere IaaS control plane está habilitado. |

6 En la página **Configurar biblioteca de contenido**, seleccione la opción **Biblioteca de contenido local** y haga clic en **Siguiente**.

Para las bibliotecas de contenido locales, importe manualmente las plantillas de OVF que desee utilizar.

7 Configure la directiva de seguridad de OVF en la página **Aplicar directiva de seguridad** y haga clic en **Siguiente** cuando haya terminado.

a Seleccione **Aplicar directiva de seguridad**.

b Seleccione la **directiva predeterminada de OVF**.

Al seleccionar esta opción, el sistema comprueba el certificado de firma de OVF durante el proceso de sincronización. Una plantilla de OVF que no supera la validación del certificado se marca con la etiqueta de **error en la verificación**. Los metadatos de la plantilla se conservan, pero los archivos OVF no se pueden sincronizar.

**Nota** Actualmente, la **directiva predeterminada de OVF** es la única directiva de seguridad compatible.

8 En la página **Agregar almacenamiento**, seleccione un almacén de datos como ubicación de almacenamiento para el contenido de la biblioteca de contenido y haga clic en **Siguiente**.

9 En la página **Listo para completar**, revise los detalles y haga clic en **Finalizar**.

10 En la página **Bibliotecas de contenido**, seleccione la nueva biblioteca de contenido que creó.

11 Descargue los archivos OVA de cada versión de Tanzu Kubernetes que desee importar a la biblioteca de contenido local.

a Visite la siguiente URL en un navegador:

https://wp-content.vmware.com/v2/latest/

b Haga clic en el directorio de la imagen que desee. Por lo general, este directorio es la última versión o la más reciente de la distribución de Kubernetes.

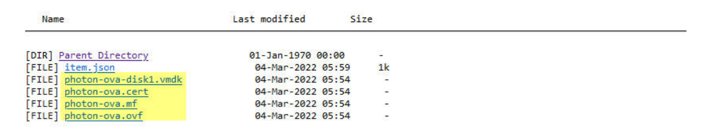
Por ejemplo:

```
ob-18186591-photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

**Importante** Deberá utilizar el nombre de distribución para importar los archivos a la biblioteca de contenido local. Debe copiar el nombre de destino en un archivo o mantener el navegador abierto hasta completar el procedimiento. La parte que se requiere de la cadena de nombre que necesitará según el ejemplo anterior es `photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067`.

c Para cada uno de los siguientes archivos, haga clic con el botón secundario y seleccione **Guardar vínculo como**.

- `photon-ova-disk1.vmdk`
- `photon-ova.cert`
- `photon-ova.mf`
- `photon-ova.ovf`




d Compruebe que cada archivo se descarga correctamente en el sistema de archivos local.

**Nota** Los archivos que importa son los archivos OVF y VMDK. Sin embargo, si se aplica una directiva de seguridad, los cuatro archivos, incluidos el certificado (`*.cert`) y el manifiesto (`*.mf`), deben estar en el directorio de origen durante la importación. Si los archivos de manifiesto y certificado no están disponibles durante la importación, la versión importada de Tanzu Kubernetes no se podrá utilizar.

12 Importe los archivos OVA y VMDK en la biblioteca de contenido local.

a Seleccione **Menú > Bibliotecas de contenido >** .

b En la lista de **Bibliotecas de contenido**, haga clic en el vínculo del nombre de la biblioteca de contenido local que haya creado.

c Haga clic en **Acciones**.

d Seleccione **Importar elemento**.

e En la ventana **Importar elemento de biblioteca**, seleccione **Archivo local**.

f Haga clic en **Cargar archivos**.

g Seleccione los archivos `photon-ova.ovf` y `photon-ova-disk1.vmdk`.

Verá el mensaje `2 files ready to import`. Cada archivo se muestra con una marca de verificación de color verde junto a su nombre.

h Cambie el nombre del **Elemento de destino** de manera que indique la versión de la imagen de OS más la versión de Kubernetes desde el directorio donde descargó los archivos.

Por ejemplo:

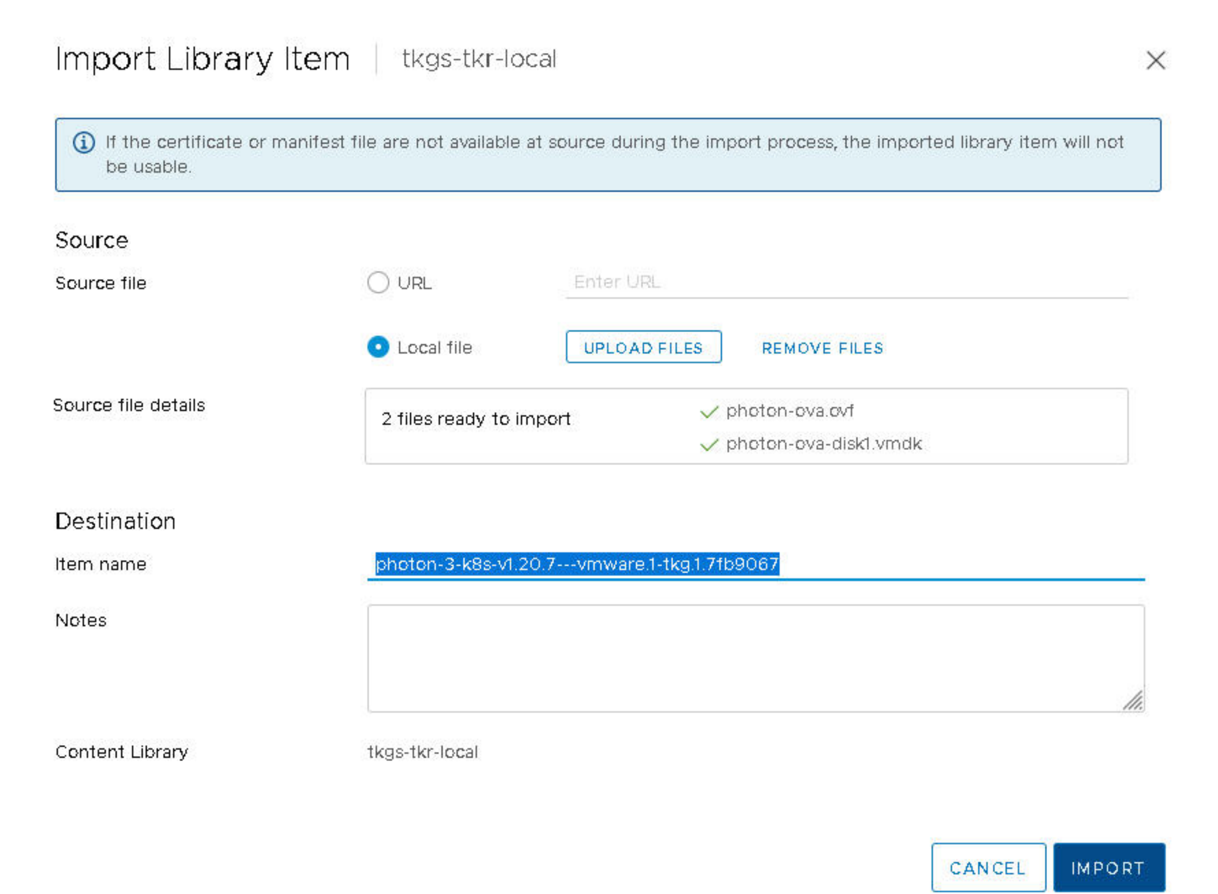
```
photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

---

**Advertencia** El nombre del **Elemento de destino** de la biblioteca de contenido debe coincidir exactamente con la cadena de nombre de carpeta para la versión deseada de Tanzu Kubernetes. Si los nombres no coinciden, Supervisor no puede resolver la imagen en una versión válida de Tanzu Kubernetes.

---

i Haga clic en **Importar**.



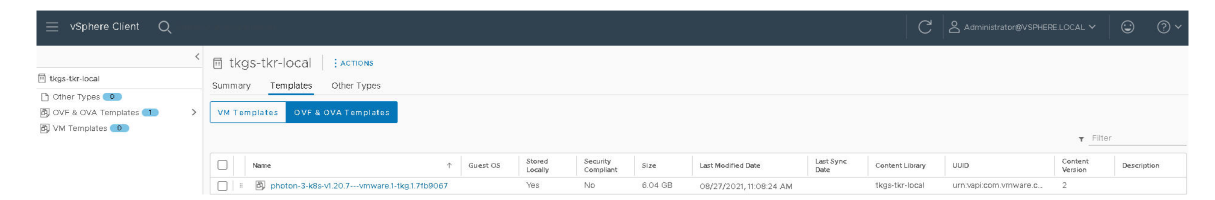
13 Compruebe que se haya rellenado la biblioteca de contenido local con la versión de Tanzu Kubernetes.

   a Despliegue el panel **Tareas recientes** en la parte inferior de la página.

   b Supervise la tarea **Obtener contenido de un elemento de biblioteca** y compruebe que se haya **Completado** correctamente.

   c En la biblioteca de contenido local, seleccione **Plantillas > Plantillas de OVF y OVA**.

   d Compruebe que los metadatos de la versión de Tanzu Kubernetes aparecen en la lista y que su contenido se almacena localmente.




### Pasos siguientes

La biblioteca de contenido de versión de Tanzu Kubernetes debe asociarse con cada espacio de nombres de vSphere donde se aprovisionen clústeres de TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

# Habilitar la publicación de una biblioteca de contenido local

Es posible crear una biblioteca de contenido local y habilitar la publicación para que otras bibliotecas de contenido se suscriban. Este enfoque permite controlar las imágenes de la versión de Tanzu Kubernetes disponibles para su uso y brindar un servicio TKR de estilo de suscripción a los operadores de clústeres de TKG.

En lugar de suscribirse a la red de entrega de contenido de VMware para versiones de Tanzu Kubernetes, donde todas las imágenes se publican y se conservan, se puede crear una biblioteca de contenido local para proporcionar un subconjunto limitado de versiones de Tanzu Kubernetes. Este enfoque resulta útil cuando se desea controlar las imágenes que se encuentran disponibles, como en los entornos de compilación continua.

### Procedimiento

1 Cree una Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados) o edite la configuración de una existente.

2 Seleccione la casilla de verificación para **Habilitar publicación**.

   Recibirá una URL de suscripción para esta biblioteca de contenido local.

3 Cree una o varias Crear una biblioteca de contenido suscrita mediante la URL de suscripción.

### Resultados

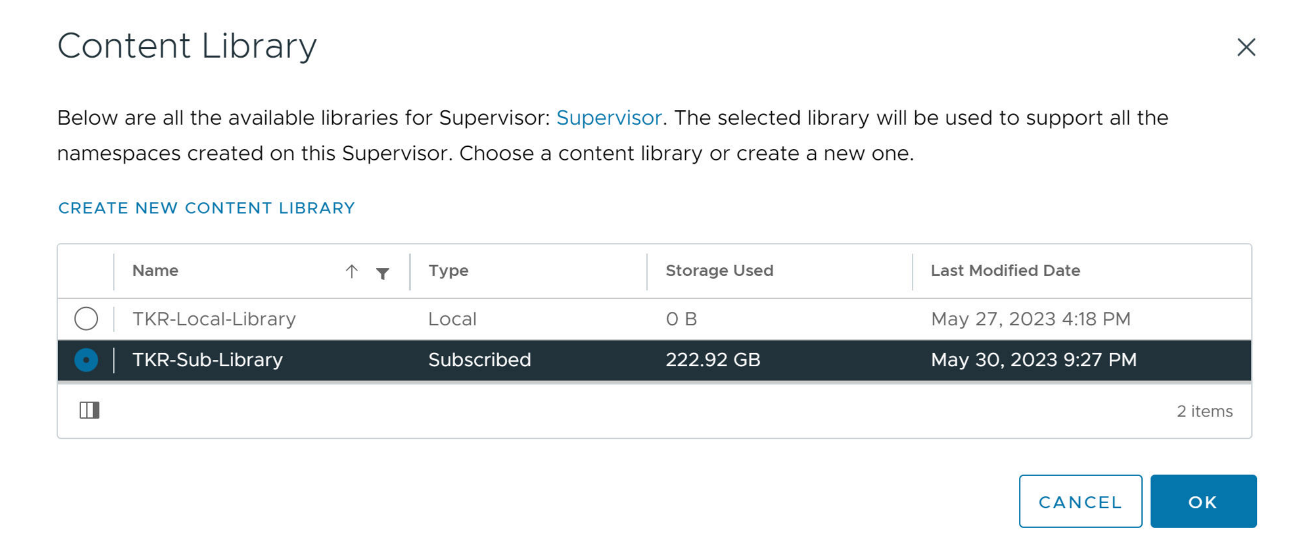
Al cargar versiones de Tanzu Kubernetes en la biblioteca de contenido local habilitada para la publicación, todas las bibliotecas de contenido suscritas recibirán las imágenes durante la sincronización. Consulte la documentación de la Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados) para obtener instrucciones sobre cómo cargar imágenes de y las Crear una biblioteca de contenido suscrita para obtener información sobre cómo crear una biblioteca de contenido suscrita.

## Editar una biblioteca de contenido existente

La biblioteca de contenido de TKR que utiliza para los clústeres de TKG no está dentro del ámbito de los espacios de nombres de vSphere individuales. Se utiliza la misma biblioteca de contenido de TKR para todos los espacios de nombres de vSphere en una instancia de Supervisor. Por este motivo, solo se permite cambiar la biblioteca de contenido de TKR en el nivel de Supervisor.

Siga estos pasos para cambiar una biblioteca de contenido de TKR existente para que la usen los espacios de nombres de vSphere.

**Figura 5-1. Editar la biblioteca de contenido de TKR para TKG**




### Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo**.

3 Seleccione la instancia de Supervisor.

4 Seleccione **Supervisor > General**.

5 Seleccione Tanzu Kubernetes Grid Service.

6 Seleccione **Biblioteca de contenido > EDITAR**.

7 Seleccione la **Biblioteca de contenido** que desea utilizar y, luego, haga clic en **ACEPTAR**.

8 Para crear una nueva biblioteca de contenido, seleccione **Crear nueva biblioteca de contenido**. Consulte Crear una biblioteca de contenido suscrita y Crear una biblioteca de contenido local (para aprovisionamiento de clústeres aislados).

Resultados

La biblioteca de contenido que seleccione se utilizará para Tanzu Kubernetes Grid Service en todas los espacios de nombres de vSphere creados en Supervisor.

# Migrar una biblioteca de contenido

Si la biblioteca de contenido de TKr alcanza su máximo de capacidad, puede migrar a una nueva biblioteca de contenido con más capacidad de almacenamiento.

Cuando el administrador de vSphere crea una biblioteca de contenido, el administrador especifica un almacén de datos donde se almacenará el contenido de la biblioteca que, en este caso, son archivos OVA. Con el paso del tiempo, a medida que se distribuyan más versiones de Kubernetes, la biblioteca de contenido se expandirá conforme se vayan agregando archivos OVA para cada actualización. Si bien no hay capacidad explícita en la biblioteca de contenido, estará limitada por la capacidad de su almacén de datos.

Si la biblioteca de contenido alcanza su capacidad, es posible que aparezca el mensaje `Internal error occurred: get library items failed for`. En ese caso, puede migrar el clúster de TKG a una nueva biblioteca de contenido para aumentar la capacidad de almacenamiento. La migración se realiza mediante vSphere Client.

Procedimiento

1 Cree una nueva biblioteca de contenido con capacidad suficiente para el clúster de destino.

2 Inicie sesión en vCenter Server mediante vSphere Client.

3 Seleccione **Menú > Hosts y clústeres**.

4 Seleccione el objeto de clúster de vSphere en el que se aprovisiona la instancia de Supervisor que contiene el clúster de Tanzu Kubernetes.

5 Seleccione la pestaña **Configurar**.

6 Seleccione la opción **Espacios de nombres > General >** en el panel de navegación.

7 Haga clic en **Editar** junto a la sección **Biblioteca de contenido** del panel principal.

8 Seleccione la nueva biblioteca de contenido que creó y haga clic en **Aceptar**.

Esta acción activa la actualización de la configuración del clúster.

**Nota** Después de modificar la biblioteca de contenido, el clúster de TKG podría tardar hasta 10 minutos en seleccionar el cambio del origen de contenido.

# Información sobre la resolución de TKr

En este tema se describe cómo el sistema resuelve las imágenes de TKr.

## Resolución de TKr

Cuando se crea o se actualiza el objeto Clúster, el servidor de la API de Kubernetes invoca el webhook mutante de TKr Resolver. El clúster (o su ClusterClass) debe tener la anotación `run.tanzu.vmware.com/resolve-tkr`. De lo contrario, la resolución de TKr se omite por completo. TKr Resolver utiliza el valor de la anotación `run.tanzu.vmware.com/resolve-tkr` como una consulta de etiqueta para restringir el conjunto de TKr candidatos. Una cadena vacía selecciona todos los TKr.

Los valores de la anotación `run.tanzu.vmware.com/resolve-os-image` en la topología del clúster controlPlain y machineDeployments se utilizan como selectores de etiquetas para los objetos OSImage que se utilizarán con controlPlain y machineDeployments respectivamente. Exactamente una OSImage enviada por el TKr resuelto debe satisfacer la consulta para controlPlane o cualquiera de los machineDeployments.

El clúster proporcionado spec.topology.version se utiliza como prefijo de versión. El webhook de TKr Resolver busca el TKr más reciente disponible que cumple las restricciones anteriores. Si no se encuentra, se deniega la solicitud de creación/actualización del clúster.

El webhook de Tkr Resolver muta el clúster:

- La etiqueta run.tanzu.vmware.com/tkr se establece como el nombre del TKr resuelto
- Cluster spec.topology.version se establece en la versión de Kubernetes del TKr resuelto
- La variable de clúster TKR_DATA se actualizó para que contenga una asignación de la versión de Kubernetes al conjunto de valores de TKr y OSImage para controlPlane.
- Las anulaciones de variables TKR_DATA para machineDeployments individuales se actualizan para que contengan una asignación de la versión de Kubernetes al conjunto de valores de TKr y OSImage para machineDeployment.

# Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG

6

Un espacio de nombres de vSphere proporciona el entorno de tiempo de ejecución para los clústeres de Servicio TKG. Para aprovisionar un clúster de Servicio TKG, primero debe configurar un espacio de nombres de vSphere con usuarios, funciones, recursos informáticos, almacenamiento, biblioteca de contenido y clases de máquinas virtuales. Los clústeres de Servicio TKG que se ejecutan en ese espacio de nombres heredan esta configuración.

Lea los siguientes temas a continuación:



## Uso de espacios de nombres de vSphere con clústeres de Servicio TKG

Un espacio de nombres de vSphere es un tenant de ámbito de red en Supervisor. Los espacios de nombres de vSphere se utilizan para alojar clústeres de Servicio TKG y proporcionan redes, permisos de función, almacenamiento persistente, cuota de recursos, biblioteca de contenido e integración de clases de máquina virtual.

## espacio de nombres de vSphere red

Una red de espacio de nombres de vSphere es una subred separada de **Supervisor > Red de carga de trabajo > Red de nombre**. El **prefijo de subred de espacio de nombres** especifica el tamaño de la subred reservada para cada espacio de nombres de vSphere. El valor predeterminado es /28.

La red de espacio de nombres de vSphere proporciona conectividad para los clústeres de TKG de Supervisor. De forma predeterminada, el espacio de nombres de vSphere utilizará configuraciones de red a nivel de clúster y asignará las direcciones IP desde su subred. Cuando se crea un espacio de nombres de vSphere, se crea una instancia del segmento de superposición de /28 y del grupo de direcciones IP correspondiente en los pods de servicio de ese espacio de nombres de vSphere

Cuando el primer clúster de TKG se aprovisiona en un espacio de nombres de vSphere, el clúster de TKG compartirá la misma subred que su espacio de nombres de vSphere. Para cada clúster de TKG subsiguiente que se aprovisiona en ese espacio de nombres de vSphere, se crea una nueva subred para ese clúster y se conecta a la puerta de enlace de su espacio de nombres de vSphere.

Existe una instancia de equilibrador de carga compartida en espacio de nombres de vSphere que se encarga de enrutar el tráfico de kubectl a cada plano de control del clúster de TKG. Además, para cada equilibrador de carga del servicio de Kubernetes que tiene recursos en el clúster de TKG, se crea una instancia de equilibrador de carga de capa 4 para ese servicio.

Los clústeres de TKG dentro del mismo espacio de nombres de vSphere comparten una IP de SNAT para la conectividad de norte a sur. La conectividad de este a oeste entre espacios de nombres no será SNAT.

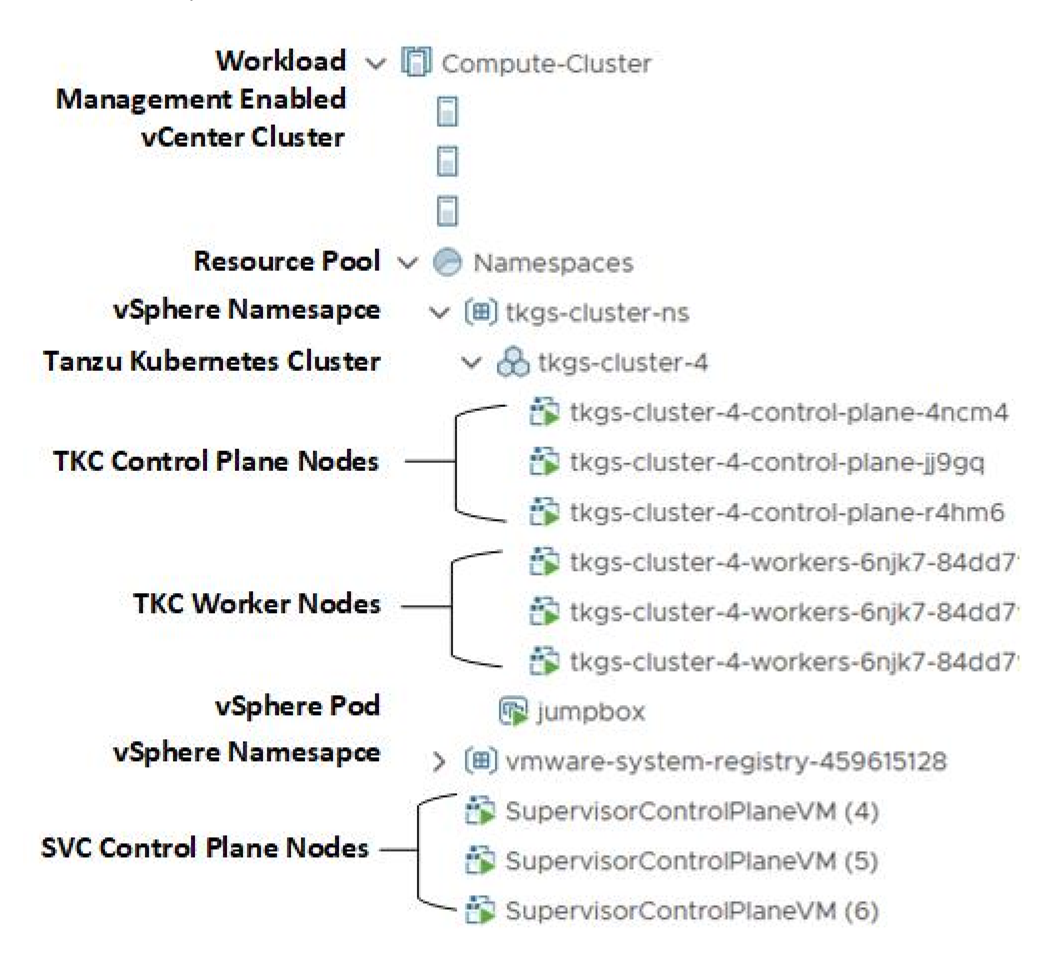
Por lo general, el espacio de nombres de vSphere no se puede enrutar. Sin embargo, si utiliza redes NSX, puede anular la red de espacio de nombres de vSphere con una subred enrutable. Consulte Anular la configuración de la red de cargas de trabajo para un espacio de nombres de vSphere.

## Grupos de recursos de espacio de nombres de vSphere

En una implementación de Supervisor de una única zona de vSphere, al crear un espacio de nombres de vSphere, se crea un grupo de recursos que respalda ese espacio de nombres. El espacio de nombres de vSphere proporciona una unidad lógica de recursos en el Supervisor, entre los que se incluyen recursos informáticos, almacenamiento, permisos, imágenes y clases. Cuando se configura un límite de CPU o memoria en un espacio de nombres de vSphere, por ejemplo, se aplican los mismos límites de recursos al grupo de recursos que respalda ese espacio de nombres. De esta manera, los espacios de nombres de vSphere habilitan el uso de varios tenants en el Supervisor.

La misma experiencia de varios tenants se aplica al Supervisor implementado entre las tres zonas de vSphere. Cuando se crea un espacio de nombres de vSphere en un Supervisor con zonas, el sistema crea un grupo de recursos en cada uno de los clústeres de vSphere que respaldan a ese Supervisor. Esto permite que un clúster de TKG aprovisionado en ese espacio de nombres de vSphere se implemente en cualquiera de las zonas que pertenecen a este Supervisor.

Si utiliza vSphere Client, puede ver objetos y el grupo de recursos del espacio de nombres de vSphere seleccionando la perspectiva **Hosts y clústeres** y la vista **Máquinas virtuales y plantillas**. Al aprovisionar un clúster de TKG, este se crea en el espacio de nombres de vSphere de destino. En una implementación de Supervisor con zonas, tendrá el mismo grupo de recursos en cada clúster de vSphere.



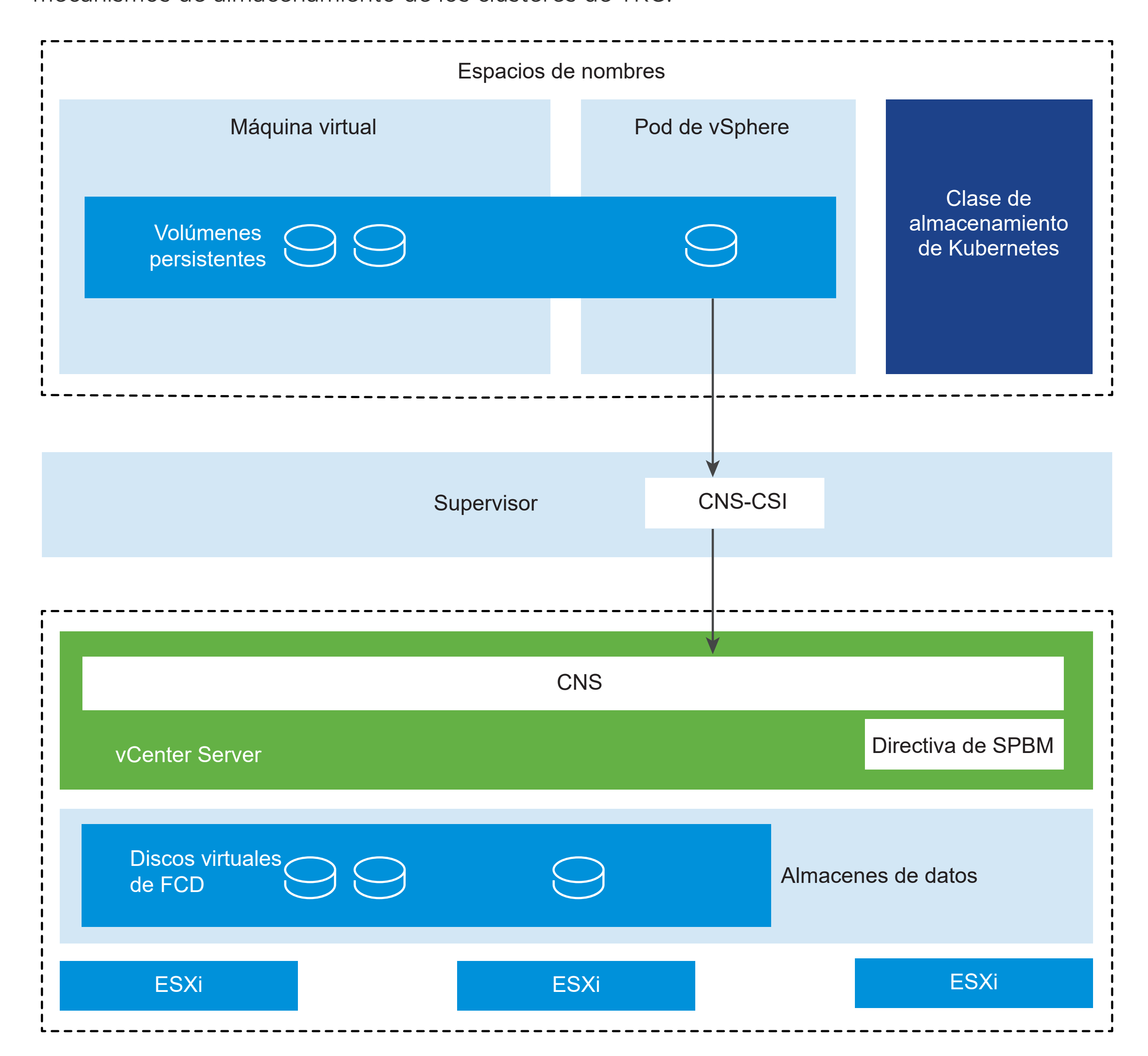

## Almacenamiento de espacio de nombres de vSphere para clústeres de Servicio TKG

El almacenamiento nativo en la nube (Cloud Native Storage, CNS) de vSphere proporciona directivas de almacenamiento que admiten el aprovisionamiento de volúmenes persistentes y sus discos virtuales de respaldo para su uso con cargas de trabajo de Kubernetes.

La interfaz de almacenamiento de contenedor (Container Storage Interface, CSI) es un estándar del sector que Kubernetes utiliza para aprovisionar almacenamiento persistente a los contenedores. El Supervisor ejecuta un controlador CNS-CSI que conecta el almacenamiento CNS de vSphere al entorno de Kubernetes a través del espacio de nombres de vSphere. El componente CNS-CSI de vSphere se comunica directamente con el plano de control de CNS para todas las solicitudes de aprovisionamiento de almacenamiento provenientes de los clústeres de TKG en el espacio de nombres de vSphere.

Los clústeres de TKG ejecutan una versión modificada del controlador CNS-CSI de vSphere que es responsable de todas las solicitudes relacionadas con el almacenamiento que se originan desde el clúster de TKG. Las solicitudes se envían al componente CNS-CSI en el Supervisor, que a su vez las propaga al CNS en vCenter Server.

El diagrama muestra la relación entre espacio de nombres de vSphere, el Supervisor y los mecanismos de almacenamiento de los clústeres de TKG.



## Volúmenes de almacenamiento persistentes para clústeres de Servicio TKG

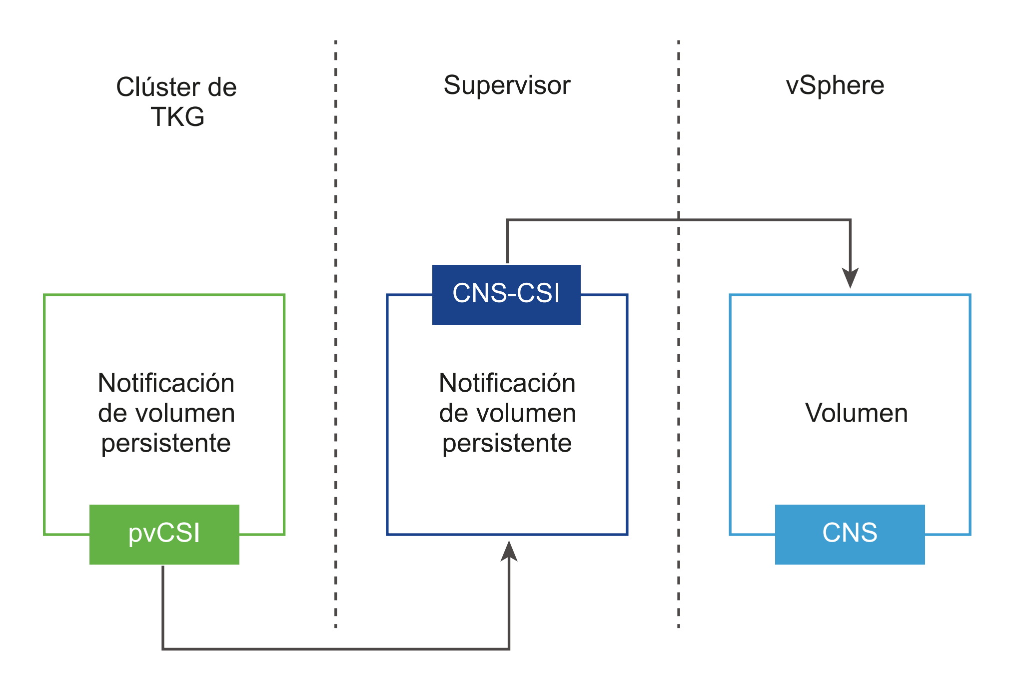
Se requieren volúmenes persistentes para las aplicaciones con estado en Kubernetes. Para obtener más información acerca de los volúmenes persistentes, consulte la documentación de Kubernetes.

En el entorno de vSphere, los objetos de volúmenes persistentes se respaldan con discos virtuales que residen en almacenes de datos. Los almacenes de datos se representan a través de directivas de almacenamiento. Cuando se asigna una directiva de almacenamiento de vSphere a un espacio de nombres de vSphere, la directiva de almacenamiento queda disponible como una clase de almacenamiento de Kubernetes para cada clúster de TKG en ese espacio de nombres.

TKG admite el aprovisionamiento dinámico y estático de volúmenes persistentes. Con el aprovisionamiento dinámico, no es necesario aprovisionar previamente el volumen persistente. Se emite una notificación de volumen persistente (Persistent Volume Claim, PVC) que hace referencia a una clase de almacenamiento disponible en el espacio de nombres de vSphere. TKG aprovisiona automáticamente el volumen persistente correspondiente con un disco virtual de respaldo. Consulte Crear volúmenes de almacenamiento persistentes de forma dinámica.

Con el aprovisionamiento estático, se utiliza un objeto de almacenamiento existente que se pone a disposición de un clúster. Usted define el volumen persistente proporcionando los detalles del objeto de almacenamiento existente, sus configuraciones admitidas y las opciones de montaje. Consulte Crear volúmenes de almacenamiento persistentes de forma estática.

El diagrama muestra el flujo de trabajo de aprovisionamiento dinámico de volúmenes persistentes. Puede crear una PVC mediante `kubectl` en el clúster de TKG. Esta acción genera una PVC coincidente en el Supervisor y activa el controlador CNS-CSI que invoca a la API de creación de volúmenes de CNS.



## Ediciones de clase de almacenamiento para clústeres de Servicio TKG

Para configurar un espacio de nombres de vSphere, asigne una o varias directivas de almacenamiento de vSphere. Cuando se aplica la directiva de almacenamiento vSphere, se convierte en una clase de almacenamiento de Kubernetes y se replica en Supervisor. Del mismo modo, la controladora TKG replica la clase de almacenamiento en cada clúster de TKG implementado en ese espacio de nombres de vSphere.

En el lado del clúster de TKG, verá dos ediciones de la clase de almacenamiento: una con el nombre definido por el usuario especificado cuando se creó la directiva de almacenamiento de vSphere y la otra con `*-latebinding` anexado al nombre.

Los desarrolladores pueden utilizar la edición de enlace en tiempo de espera de la clase de almacenamiento para enlazarse a un volumen de almacenamiento persistente después de que el programador de pods de TKG seleccione el nodo informático. Para obtener más información sobre qué clase de almacenamiento usar y cuándo, consulte Usar clases de almacenamiento para volúmenes persistentes.

```
kubectl get sc
NAME                                    PROVISIONER              RECLAIMPOLICY
VOLUMEBINDINGMODE       ALLOWVOLUMEEXPANSION   AGE
wcpglobal-storage-profile               csi.vsphere.vmware.com   Delete
Immediate               true                   2m43s
wcpglobal-storage-profile-latebinding   csi.vsphere.vmware.com   Delete
WaitForFirstConsumer    true                   2m43s
```

## Creación de un espacio de nombres de vSphere

Existen varias formas de crear espacios de nombres de vSphere.

Los administradores pueden crear espacios de nombres de vSphere mediante vSphere Client. Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

Los usuarios de vCenter Single Sign-On a los que se concede el permiso del rol **Propietario** para un espacio de nombres de vSphere pueden crear espacios de nombres de vSphere en modo de autoservicio mediante `kubectl`. Consulte Habilitar la creación de espacio de nombres de vSphere mediante Kubectl.

VMware expone API de vCenter Server para administrar el ciclo de vida de espacios de nombres de vSphere y proporciona kits de desarrollo de software (SDK), entre los que se incluyen:

- Las API de administración de espacios de nombres proporcionan recursos basados en REST para administrar espacios de nombres de vSphere.
- Las API de espacios de nombres proporcionan recursos basados en REST para administrar el control de acceso de sujetos en espacios de nombres de vSphere.
- El vSphere Automation SDK para Java proporciona varios paquetes para automatizar la creación de espacios de nombres de vSphere y la administración de su ciclo de vida.

- Del mismo modo, el vSphere Automation SDK para Python proporciona varios paquetes para automatizar la creación de espacios de nombres de vSphere y la administración de su ciclo de vida.

# Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG

Los clústeres de Servicio TKG se aprovisionan en espacio de nombres de vSphere.

Cree un espacio de nombres de vSphere para alojar uno o varios clústeres de Servicio TKG.

### Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo > Espacios de nombres > Nuevo espacio de nombres**.

3 Seleccione el clúster de vSphere en el que se habilitó el Supervisor.

4 Introduzca un nombre para espacio de nombres de vSphere.

  El nombre debe tener un formato conforme a DNS y debe ser único en todas las instancias de Supervisor administradas por vCenter Server.

5 Introduzca una descripción para el espacio de nombres de vSphere.

6 Especifique la **Red de espacio de nombres** según el tipo de pila de red utilizada para el Supervisor.

| Pila de red | |
|---|---|
| **Redes de vSphere** | Seleccione una red de carga de trabajo del menú **Red**. |
| **Redes NSX** | Herede la red del Supervisor, en cuyo caso no se necesita ninguna acción.<br>O bien, seleccione **Anular configuración de red de clúster** y personalice la red para este espacio de nombres de vSphere. Consulte Anular la configuración de la red de cargas de trabajo para un espacio de nombres de vSphere. |

7 Haga clic en **Crear** para crear el espacio de nombres de vSphere.

  El espacio de nombres se crea en el Supervisor.

8 Configure el espacio de nombres de vSphere para los clústeres de TKG. Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

# Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG

Puede implementar uno o varios clústeres de Servicio TKG en un espacio de nombres de vSphere. Cada clúster de Servicio TKG implementado allí hereda las opciones de configuración aplicadas al espacio de nombres de vSphere.

## Configurar permisos de función para el espacio de nombres de vSphere

Los permisos de función se limitan al ámbito del espacio de nombres de vSphere. Hay tres permisos de función que se pueden asignar a los usuarios y grupos del clúster de TKG: **Propietario**, **Puede editar** y **Puede ver**. En la tabla se describe cada función. Para obtener más información, consulte Acerca de la administración de identidades y acceso para clústeres de Servicio TKG.

Si utiliza vCenter Single Sign-On, las tres funciones estarán disponibles. Para asignar usuarios y grupos de SSO a un espacio de nombres de vSphere, consulte Configurar los permisos del espacio de nombres de vSphere para usuarios y grupos de vCenter Single Sign-On.

Si utiliza un proveedor de OIDC externo, la función **Propietario** no estará disponible. Para asignar usuarios y grupos de OIDC a un espacio de nombres de vSphere, consulte Configurar permisos de espacio de nombres de vSphere para usuarios y grupos del proveedor de identidad externo.

| Función | Descripción |
| --- | --- |
| Propietario | El permiso de función Enlaces y permisos de función permite a los usuarios y grupos asignados administrar objetos de espacio de nombres de vSphere mediante kubectl y utilizar clústeres de TKG. Consulte Habilitar la creación de espacio de nombres de vSphere mediante Kubectl. |
| Puede editar | El permiso de función Enlaces y permisos de función permite a los usuarios y grupos asignados ver objetos de espacio de nombres de vSphere y utilizar clústeres de TKG.<br>Un grupo o un usuario de vCenter Single Sign-On con el permiso **Puede editar** está enlazado a la función **cluster-admin** de Kubernetes para cada clúster de TKG implementado en esa instancia de espacio de nombres de vSphere. |
| Puede ver | El permiso de función Enlaces y permisos de función permite a los usuarios y grupos asignados ver objetos de espacio de nombres de vSphere.<br>**Nota** No hay ninguna función equivalente de solo lectura en Kubernetes a la que se pueda enlazar el permiso **Puede ver**. Para conceder acceso al clúster a los usuarios de Kubernetes, consulte Conceder a los desarrolladores acceso de vCenter SSO a los clústeres de Servicio TKG. |

## Configurar el almacenamiento persistente para el espacio de nombres de vSphere

Puede asignar una o varias directivas de almacenamiento de vSphere a un espacio de nombres de vSphere. Una directiva de almacenamiento asignada controla la colocación de almacenes de datos de volúmenes persistentes en el entorno de almacenamiento de vSphere.

Por lo general, un administrador de vSphere define una directiva de almacenamiento vSphere. Si utiliza zonas de vSphere, la directiva de almacenamiento debe configurarse con la topología `Zonal`. Consulte Crear una directiva de almacenamiento de vSphere para clústeres de Servicio TKG.

Para asignar una directiva de almacenamiento de vSphere a un espacio de nombres de vSphere:

1 Seleccione **Administración de cargas de trabajo > Espacio de nombres** y el espacio de nombres de vSphere de destino.

2 En el mosaico **Almacenamiento**, seleccione **Agregar almacenamiento**.

3 Seleccione una o varias directivas de almacenamiento entre las opciones disponibles.

Para cada directiva de almacenamiento de vSphere que se asigna a un espacio de nombres de vSphere, el sistema crea dos clases de almacenamiento de Kubernetes coincidentes en ese espacio de nombres de vSphere. Estas clases de almacenamiento se replican en cada clúster de TKG implementado en ese espacio de nombres de vSphere. Consulte Usar clases de almacenamiento para volúmenes persistentes.

## Establecer límites de capacidad y uso para el espacio de nombres de vSphere

Al configurar un espacio de nombres de vSphere, se crea un grupo de recursos para el espacio de nombres de vSphere en vCenter Server. De forma predeterminada, este grupo de recursos está configurado sin capacidad ni cuota de uso; los recursos están limitados por la infraestructura.

En el mosaico **Capacidad y uso** del espacio de nombres de vSphere, puede configurar los siguientes **Límites**.

| | |
|---|---|
| CPU | La cantidad de recursos de CPU que se reservarán para el espacio de nombres de vSphere. |
| Memoria | La cantidad de memoria que se reservará para el espacio de nombres de vSphere. |
| Almacenamiento | La cantidad total de espacio de almacenamiento que se reservará para el espacio de nombres de vSphere. |
| Límite de directiva de almacenamiento | Establezca la cantidad de almacenamiento dedicado individualmente a cada una de las directivas de almacenamiento que están asociadas al espacio de nombres de vSphere. |

Por lo general, para las implementaciones de clústeres de TKG, no es necesario configurar la cuota de recursos en el espacio de nombres de vSphere. Si asigna límites de cuota, es importante conocer cuál sería el posible impacto en los clústeres de TKG implementados allí.

**Límites de CPU y memoria**

Los límites de CPU y memoria configurados en el espacio de nombres de vSphere no tienen ninguna relación con un clúster de TKG implementado allí si los nodos del clúster de TKG utilizan el tipo de clase de máquina virtual garantizado. Sin embargo, si los nodos del clúster de TKG utilizan el Uso de clases de máquinas virtuales con clústeres de Servicio TKG, los límites de CPU y memoria pueden afectar al clúster de TKG.

Como el tipo de clase de máquina virtual de mejor esfuerzo permite la sobreasignación de recursos, es posible quedarse sin recursos si se establecieron límites de CPU y memoria en el espacio de nombres de vSphere donde se aprovisionan los clústeres de TKG. Si se produce una contención y el plano de control del clúster de TKG se ve afectado, el clúster puede dejar de ejecutarse. Por este motivo, siempre debe utilizar el Uso de clases de máquinas virtuales con clústeres de Servicio TKG para los clústeres de producción. Si no puede usar el tipo de clase de máquina virtual garantizada para todos los nodos de producción, debe usarlo como mínimo para los nodos del plano de control.

**Almacenamiento y límites de directivas de almacenamiento**

Un límite de almacenamiento configurado en el espacio de nombres de vSphere determina la cantidad total de almacenamiento que está disponible en el espacio de nombres de vSphere para todos los clústeres de TKG implementados allí.

Un límite de directiva de almacenamiento configurado en el espacio de nombres de vSphere determina la cantidad de almacenamiento disponible en esa clase de almacenamiento para cada clúster de TKG donde se replica la clase de almacenamiento.

Algunas cargas de trabajo tienen requisitos mínimos de almacenamiento. Consulte #unique_112, por ejemplo.

## Asociar la biblioteca de contenido de TKR al Servicio TKG

Para aprovisionar clústeres de TKG, asocie Servicio TKG a la biblioteca de contenido. Para crear una biblioteca de contenido a fin de alojar imágenes de TKr, consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

Para asociar una biblioteca de contenido de TKr con un espacio de nombres de vSphere:

1 Seleccione **Administración de cargas de trabajo > Supervisores > Supervisor** y seleccione la instancia de Supervisor.

2 Seleccione **Configurar > Supervisor > General > Tanzu Kubernetes Grid Service**.

3 Seleccione **Biblioteca de contenido > Editar**.

4 Seleccione una biblioteca de contenido de TKR.

5 Vaya a espacio de nombres de vSphere y seleccione **Administrar espacio de nombres**.

6 Compruebe que la biblioteca de contenido seleccionada aparezca en el panel de configuración de **Tanzu Kubernetes Grid Service**.

Es importante comprender que la biblioteca de contenido de TKR no está en el ámbito del espacio de nombres. Todas las instancias de espacios de nombres de vSphere usan la misma biblioteca de contenido de TKR que está configurada para el servicio Tanzu Kubernetes Grid Service (TKGS). La edición de la biblioteca de contenido de TKR para TKGS se aplicará a cada instancia de espacio de nombres de vSphere.

**Nota** La biblioteca de contenido a la que se hace referencia en el mosaico de **Servicio de máquina virtual** debe usarse con máquinas virtuales independientes y no con la biblioteca de contenido de TKR. No agregue la biblioteca de contenido de TKR a este mosaico.

## Asociar las clases de máquinas virtuales con el espacio de nombres de vSphere

vSphere IaaS control plane proporciona varias Uso de clases de máquinas virtuales con clústeres de Servicio TKGpredeterminadas y puede crear las suyas propias.

Para aprovisionar un clúster de TKG, asocie una o varias Uso de clases de máquinas virtuales con clústeres de Servicio TKG con el espacio de nombres de vSphere de destino. Las clases enlazadas están disponibles para que las usen los nodos del clúster de TKG implementados en ese espacio de nombres de vSphere.

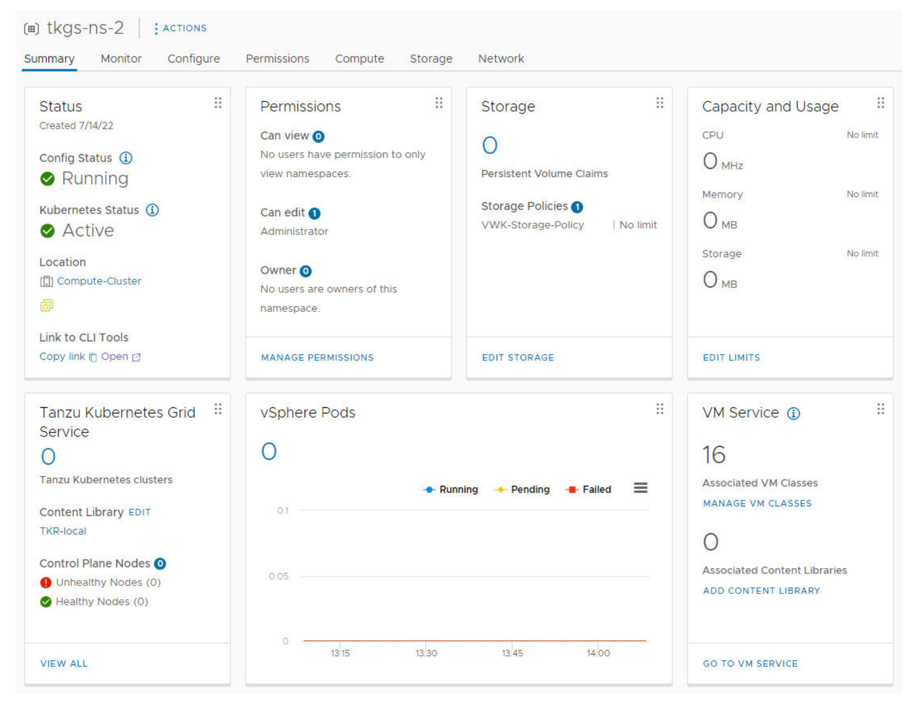
Para asociar las clases de máquinas virtuales predeterminadas con un espacio de nombres de vSphere, inicie sesión en vCenter Server mediante vSphere Client y complete el siguiente procedimiento.

1. Seleccione **Administración de cargas de trabajo > Espacio de nombres** y el espacio de nombres de vSphere de destino.
2. En el mosaico **Servicio de máquina virtual**, seleccione **Agregar clase de máquina virtual**.
3. Seleccione cada una de las clases de máquinas virtuales que desea agregar.
   a. Para agregar las clases de máquina virtual predeterminadas, active la casilla de verificación del encabezado de la tabla en la página 1 de la lista, desplácese hasta la página 2 y seleccione la casilla de verificación en el encabezado de la tabla de esa página. Compruebe que todas las clases estén seleccionadas.
   b. Para crear una clase personalizada, haga clic en **Crear nueva clase de máquina virtual**. Consulte la documentación de Servicios de máquina virtual para obtener instrucciones.
4. Haga clic en **Aceptar** para completar esta operación.
5. Confirme que se hayan agregado las clases. El mosaico de **Servicio de máquina virtual** muestra **Administrar clases de máquinas virtuales**.

## Comprobar la configuración del espacio de nombres de vSphere

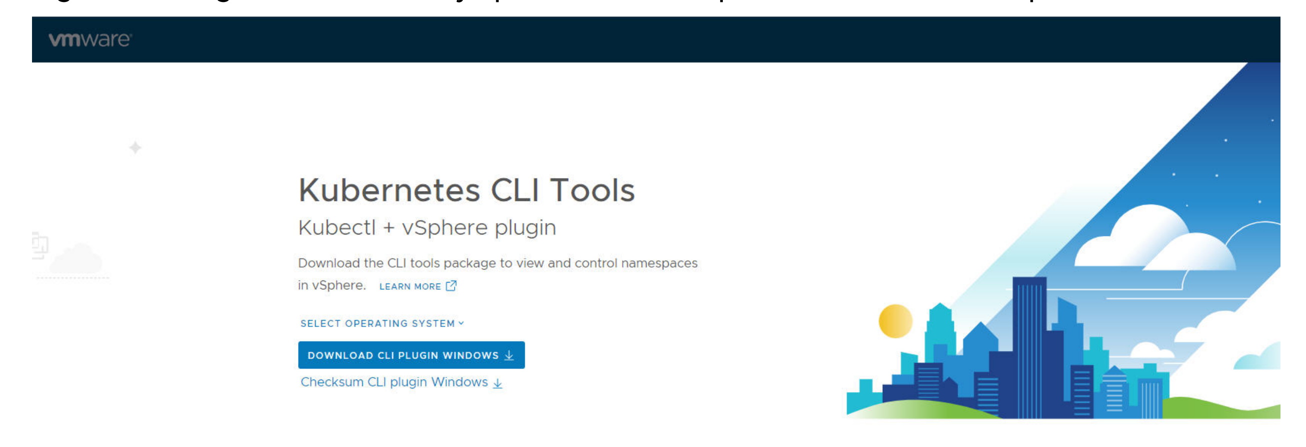
Un espacio de nombres de vSphere configurado incluye **Estado**, **Permisos**, **Almacenamiento**, **Capacidad y uso**, **Biblioteca de contenido de TKR** y **Clases de máquina virtual**.

Figura 6-1. espacio de nombres de vSphere (configurado)




El mosaico **Estado** incluye un vínculo a las herramientas de la CLI de vSphere IaaS control plane. El equilibrador de carga del plano de control de Supervisor utiliza la página de desarrollo y operaciones. Proporcione el vínculo a los usuarios del clúster de TKG para descargar las Herramientas de la CLI de Kubernetes para vSphere. Consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

Figura 6-2. Página de desarrollo y operaciones del espacio de nombres de vSphere




Para comprobar la configuración del espacio de nombres de vSphere mediante kubectl, consulte Comprobar la preparación de espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

# Anular la configuración de la red de cargas de trabajo para un espacio de nombres de vSphere

Cuando se implementa Supervisor, se deben establecer las opciones de configuración predeterminadas para la red de cargas de trabajo. Si habilitó Supervisor con redes de NSX, puede anular las opciones de configuración predeterminadas de la red de cargas de trabajo mediante la creación de un espacio de nombres de vSphere. Las opciones de configuración anuladas de la red de cargas de trabajo solo se aplican a este segmento del espacio de nombres de vSphere.

## Anular la configuración de la red de cargas de trabajo (solo NSX)

Al crear un espacio de nombres de vSphere, se crea un segmento de red. De forma predeterminada, este segmento de red se deriva de la red de cargas de trabajo configurada en el Supervisor. Consulte espacio de nombres de vSphere red para obtener más información.

Si el Supervisor está configurado con redes NSX, durante la creación de un espacio de nombres de vSphere, tendrá la opción de elegir **Anular configuración de red de clúster** para el espacio de nombres de vSphere. Al elegir esta opción, se podrá personalizar la red del espacio de nombres de vSphere, para lo que tendrá que agregar los CIDR a los campos Entrada, Salida y Red de espacio de nombres. Los nuevos CIDR que agregue reemplazarán los CIDR existentes para esta instancia del espacio de nombres de vSphere.

Si configuró NSX versión 4.1.1 o posterior e instaló, configuró y registró NSX Advanced Load Balancer versión 22.1.4 o posterior con una licencia Enterprise en NSX, el equilibrador de carga que se utiliza con NSX es NSX Advanced Load Balancer. Si configuró versiones de NSX anteriores a la 4.1.1, se utiliza el equilibrador de carga de NSX. Para obtener más información, consulte Redes de supervisor en *Planificación y conceptos del plano de control de IaaS de vSphere*.

El caso práctico típico para anular la configuración de red de Supervisor es aprovisionar un clúster de TKG con redes de pods enrutables. Consulte las opciones de configuración de la tabla para obtener más información sobre cómo hacerlo y obtener vínculos a ejemplos.

### Tabla 6-1. Consideraciones sobre la planificación de la red de espacio de nombres de vSphere

| Consideración | Descripción |
| --- | --- |
| Se requiere NSX | Para anular la configuración de red del Supervisor para un espacio de nombres de vSphere en particular, el Supervisor debe configurarse con redes de NSX. |
| Instalación de NSX | A fin de anular la configuración de red de Supervisor para una instancia de espacio de nombres de vSphere en particular, la instalación de NSX debe incluir un clúster de Edge dedicado para puertas de enlace de nivel 0 (enrutadores) y otro clúster de Edge dedicado para puertas de enlace de nivel 1. Consulte la guía de instalación de NSX *Instalar y configurar vSphere with Tanzu*. |

### Tabla 6-1. Consideraciones sobre la planificación de la red de espacio de nombres de vSphere (continuación)

| Consideración | Descripción |
| --- | --- |
| Se requiere IPAM | Si anula la configuración de red del Supervisor para un espacio de nombres de vSphere en particular, la nueva red del espacio de nombres de vSphere debe especificar las subredes de Entrada, Salida y Red de espacio de nombres que sean únicas en el Supervisor y en cualquier otra red del espacio de nombres de vSphere. Tendrá que administrar la asignación de direcciones IP según corresponda. |
| Enrutamiento del supervisor | Supervisor debe poder enrutar directamente a los nodos del clúster de TKG y a las subredes de entrada. Al seleccionar una puerta de enlace de nivel 0 para el espacio de nombres de vSphere, existen dos opciones para configurar el enrutamiento que se requiere:<br>■ Utilizar una puerta de enlace de enrutamiento y reenvío virtual (Virtual Routing and Forwarding, VRF) para heredar la configuración de la puerta de enlace de nivel 0 del Supervisor<br>■ Utilizar el protocolo de puerta de enlace de borde (Border Gateway Protocol, BGP) para configurar rutas entre la puerta de enlace de nivel 0 del Supervisor y la puerta de enlace de nivel 0 dedicada<br>Consulte la documentación de las puertas de enlace de nivel 0 de NSX para conocer más detalles sobre estas opciones. |

Campos de configuración para anular la configuración de red del Supervisor.

### Tabla 6-2. Opciones de configuración del espacio de nombres de vSphere para anular la configuración de la red de cargas de trabajo

| Componente | Configuración |
| --- | --- |
| Puerta de enlace de nivel 0 | La puerta de enlace de nivel 0 de NSX conecta Supervisor a la red física. La puerta de enlace de nivel 0 seleccionada se asociará con la puerta de enlace de nivel 1 que se creará para el espacio de nombres de vSphere.<br>Al seleccionar una nueva puerta de enlace de nivel 0, se anula la puerta de enlace de nivel 0 configurada cuando se habilitó el Supervisor. En este caso, debe configurar nuevos rangos de CIDR. Si selecciona una puerta de enlace VRF vinculada a la puerta de enlace de nivel 0, la red y las subredes se configurarán automáticamente.<br>Una vez que se selecciona una puerta de enlace de nivel 0 y se completa su configuración, no se puede cambiar la puerta de enlace de nivel 0. |
| Tamaño del equilibrador de carga | Seleccione el tamaño de la instancia del equilibrador de carga en la puerta de enlace de nivel 1 para el espacio de nombres de vSphere.<br>Establezca el tamaño del equilibrador de carga en pequeño (valor predeterminado), mediano o grande. Solo se puede definir un número establecido de instancias de equilibrador de carga por nodo de Edge. Consulte la configuración máxima para obtener detalles.<br>**Nota** Esta configuración no es aplicable para el NSX Advanced Load Balancer. |

## Tabla 6-2. Opciones de configuración del espacio de nombres de vSphere para anular la configuración de la red de cargas de trabajo (continuación)

| Componente | Configuración |
| --- | --- |
| Modo NAT | El modo NAT está seleccionado de forma predeterminada. Esto significa que se espera que la subred de la red de espacio de nombres no se pueda enrutar, y debe configurar los CIDR de red de espacio de nombres, entrada y salida.<br>Al anular la selección del modo NAT, se indica al sistema que se va a proporcionar un rango de CIDR enrutable para la red de espacios de nombres. Si anula la selección del modo de NAT, las direcciones IP del nodo de clústeres de TKG son accesibles directamente desde fuera de la puerta de enlace de nivel 0 y no es necesario configurar los CIDR de salida.<br>Para aprovisionar un clúster sin modo NAT, anule la selección del modo NAT y consulte los ejemplos: Capítulo 7 Aprovisionar clústeres del servicio de TKG. |
| CIDR de red de espacio de nombres | El CIDR de red de espacio de nombres es una subred que funciona como un grupo de direcciones IP en la que el prefijo de subred de espacio de nombres describe el tamaño de cualquier bloque CIDR subsiguiente separado de ese grupo de direcciones IP.<br>Cada vez que se crea un espacio de nombres de vSphere, se asigna una subred de la red de espacio de nombres. El tamaño de subred separado de este bloque es /24; lo que significa que se puede crear un máximo de 256 pods por espacio de nombres de vSphere. Hacer referencia al máx. de config. para obtener detalles.<br>El CIDR de red de espacio de nombres se utiliza para asignar direcciones IP a clústeres de TKG asociados a segmentos del espacio de nombres de vSphere.<br>Si se selecciona el modo NAT, se espera que el CIDR no se pueda enrutar. Si el modo NAT no está marcado, el CIDR de red de espacio de nombres debe poder enrutarse. |
| Prefijo de subred de espacio de nombres | Introduzca el prefijo de subred que especifica el tamaño de la subred reservada para los segmentos de espacios de nombres. El valor predeterminado es 28.<br>El prefijo de subred de espacio de nombres define la subred IP creada para cada segmento del espacio de nombres de vSphere. Por ejemplo, si se establece un prefijo /24, se generará un segmento del espacio de nombres de vSphere con una subred IP de 254 direcciones IP para asignar a los clústeres de TKG implementados allí.<br>Otro ejemplo:<br>CIDR de red de espacio de nombres= 192.168.1.0/24<br>Prefijo de subred de espacio de nombres= /28<br>En este caso, TKGS puede proporcionar 16 bloques CIDR de 192.168.1.x/28 desde la subred 192.168.1.0/24. Esto permite crear instancias de 16 redes de espacio de nombres de TKGS a las que están conectadas las máquinas virtuales administradas por Servicio TKG (TKC, VMS, pods de vSphere). Por ejemplo, cada TKC recibe un CIDR de espacio de nombres dedicado y, en este caso, podría ser 192.168.1.0/28 y la siguiente subred de espacio de nombres de TKC sería 192.168.0.16/28 y así sucesivamente. |

**Tabla 6-2. Opciones de configuración del espacio de nombres de vSphere para anular la configuración de la red de cargas de trabajo (continuación)**

| Componente | Configuración |
| --- | --- |
| Entrada | El bloque CIDR de IP de entrada se utiliza para asignar direcciones IP para servicios de Kubernetes publicados por un equilibrador de carga de tipo de servicio y por un controlador de entrada en todos los espacios de nombres de vSphere. Los servicios de clúster de TKG y la entrada obtienen sus direcciones IP de este bloque de CIDR.<br>Introduzca una anotación CIDR que determine el rango de IP de entrada para las direcciones IP virtuales publicadas por el servicio de equilibrador de carga para los clústeres de TKG.<br>**Nota** Esta configuración no es aplicable para el NSX Advanced Load Balancer. |
| Salida | Se usa una CIDR de IP de salida para asignar direcciones IP para la Traducción de Direcciones de Red de Origen (Source Network Address Translation, SNAT) para el tráfico que sale del espacio de nombres de vSphere para acceder a servicios externos.<br>Introduzca una anotación CIDR que determine el rango de IP de salida para las direcciones IP SNAT. |

# Uso de clases de máquinas virtuales con clústeres de Servicio TKG

Para cambiar nodos del clúster de Servicio TKG, especifique la clase de máquina virtual (VM). La plataforma proporciona clases de VM predeterminadas y es posible crear clases propias. Para utilizar una clase de máquina virtual, puede asociarla con la instancia de espacio de nombres de vSphere de destino y hacer referencia a la clase en el manifiesto del clúster.

## Acerca de las clases de máquina virtual

Una clase de máquina virtual es una solicitud de reservas de recursos para potencia de procesamiento en la máquina virtual, incluidas la CPU y la memoria (RAM). Por ejemplo: el tipo de clase de máquina virtual denominado "guaranteed-large" reserva 4 CPU y 16 GB de RAM.

**Nota** El tamaño de disco de la máquina virtual se establece mediante la plantilla de OVA, no la definición de clase de máquina virtual. Para versiones de Tanzu Kubernetes, el tamaño de disco es de 16 GB.

Existen dos tipos de reserva para las clases de máquina virtual: garantizada y mejor esfuerzo. La clase garantizada reserva por completo los recursos configurados. Esto significa que, para un clúster determinado, `spec.policies.resources.requests` coincide con la configuración de `spec.hardware`. La clase de mejor esfuerzo permite que se genere un compromiso excesivo de los recursos. Para las cargas de trabajo de producción se recomienda utilizar el tipo de clase de máquina virtual garantizada.

**Advertencia** Debido a que el tipo de clase de máquina virtual de mejor esfuerzo permite la sobreasignación, es posible quedarse sin recursos si se establecieron límites en el espacio de nombres de vSphere donde está aprovisionando el clúster de TKG. Si se produce una contención y el plano de control se ve afectado, el clúster puede dejar de ejecutarse. Por este motivo, utilice el tipo de clase de máquina virtual garantizada para los clústeres de producción. Si no puede usar el tipo de clase de máquina virtual garantizada para todos los nodos de producción, utilícelo como mínimo para los nodos del plano de control.

## Uso de clases de máquinas virtuales con clústeres de Servicio TKG

Para usar una clase de máquina virtual con un clúster de servicio TKG, la clase de máquina virtual debe estar enlazada con la instancia de espacio de nombres de vSphere donde se aprovisiona el clúster. Para ello, asocie la clase con el espacio de nombres de destino. Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

Para enumerar las clases de máquina virtual disponibles en el espacio de nombres de vSphere de destino, use el comando `kubectl get virtualmachineclass`.

**Nota** Si tiene problemas con este comando, consulte Corregir errores de clase de máquina virtual.

Las definiciones de clase de máquina virtual no son inmutables. Todas las clases de máquina virtual se pueden editar, incluidas las definiciones predeterminadas de clases de máquina virtual. Si se edita una clase de máquina virtual, los nodos del clúster de TKG existentes no se ven afectados. Los nuevos clústeres de TKG utilizan la clase modificada.

**Precaución** Si edita una clase de máquina virtual que está utilizando un clúster de TKG y, a continuación, escala horizontalmente ese clúster, los nodos nuevos utilizarán la definición de clase editada, pero los nodos existentes usarán la definición de clase inicial, lo que provocará un error de coincidencia de clases.

## Clases de máquina virtual predeterminadas

La tabla muestra los tipos de clase de máquina virtual predeterminados que se utilizan como tamaños de implementación para los nodos del clúster de Tanzu Kubernetes.

Para evitar la sobreasignación de recursos, las cargas de trabajo de producción deben utilizar el tipo de clase garantizado. Para evitar quedarse sin memoria, no utilice el tamaño de clase pequeño o muy pequeño para ningún nodo de trabajo en el que implemente cargas de trabajo en cualquier entorno (desarrollo, prueba o producción).

Tabla 6-3. Clases de máquina virtual predeterminadas

| Clase | CPU | Memoria (GB) | CPU y memoria reservadas |
| --- | --- | --- | --- |
| guaranteed-8xlarge | 32 | 128 | Sí |
| best-effort-8xlarge | 32 | 128 | No |
| guaranteed-4xlarge | 16 | 128 | Sí |
| best-effort-4xlarge | 16 | 128 | No |
| guaranteed-2xlarge | 8 | 64 | Sí |
| best-effort-2xlarge | 8 | 64 | No |
| guaranteed-xlarge | 4 | 32 | Sí |
| best-effort-xlarge | 4 | 32 | No |
| guaranteed-large | 4 | 16 | Sí |
| best-effort-large | 4 | 16 | No |
| guaranteed-medium | 2 | 8 | Sí |
| best-effort-medium | 2 | 8 | No |
| guaranteed-small | 2 | 4 | Sí |
| best-effort-small | 2 | 4 | No |
| guaranteed-xsmall | 2 | 2 | Sí |
| best-effort-xsmall | 2 | 2 | No |

## Clases de máquina virtual personalizadas

vSphere IaaS control plane admite clases de máquina virtual personalizadas para usarlas con clústeres de TKG. Una vez que haya definido una clase de máquina virtual personalizada, debe asociarla con el espacio de nombres de vSphere de destino para poder utilizarla con un clúster. Consulte más detalles en la documentación de Servicios de máquina virtual.

# Comprobar la preparación de espacio de nombres de vSphere para alojar clústeres de Servicio TKG

Una vez que haya configurado un espacio de nombres de vSphere, inicie sesión en el Supervisor y compruebe que el espacio de nombres de vSphere esté listo para aprovisionar clústeres de Servicio TKG.

## Comprobar la configuración de espacio de nombres de vSphere para el aprovisionamiento de clústeres de servicio TKG

Para preparar el aprovisionamiento de clústeres de servicio TKG, complete esta tarea a fin de comprobar que el espacio de nombres de vSphere esté configurado correctamente.

1 Inicie sesión en el Supervisor.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere de destino en el que tiene pensado aprovisionar uno o varios clústeres de TKG.

```
kubectl config use-context VSPHERE-NAMESPACE-NAME
```

3 Describa el espacio de nombres de vSphere.

```
kubectl describe ns VSPHERE-NAMESPACE-NAME
```

Este comando devuelve el nombre de cada clase de almacenamiento disponible en el espacio de nombres de vSphere, así como la cuota de recursos.

4 Enumere y describa las versiones de Tanzu Kubernetes disponibles.

```
kubectl get tanzukubernetesreleases
```

Este comando devuelve las TKR que se encuentran en la biblioteca de contenido configurada para el espacio de nombres de vSphere y que se sincronizaron con la biblioteca o se cargaron en esta.

5 Enumere las clases de máquinas virtuales disponibles.

```
kubectl get virtualmachineclass
```

Este comando devuelve las clases de máquina virtual que están asociadas al espacio de nombres. Solo se pueden utilizar clases de máquina virtual enlazadas para aprovisionar nodos de clúster de servicio TKG.

# Habilitar la creación de espacio de nombres de vSphere mediante Kubectl

Es posible habilitar el servicio de espacio de nombres de vSphere para permitir que los desarrolladores administren el ciclo de vida de espacios de nombres de vSphere mediante `kubectl`.

Al habilitar el servicio de espacio de nombres de vSphere en Supervisor, los desarrolladores asignados a la función Propietario en un espacio de nombres de vSphere pueden crear sus propios espacios de nombres de vSphere mediante el comando `kubectl create namespace <NAME>`.

Cuando se habilita el servicio de espacio de nombres de vSphere, se define y se activa una plantilla de espacio de nombres. Los desarrolladores a los que se asigna la función Propietario utilizan la plantilla para crear espacios de nombres.

Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo > Supervisores** y seleccione la instancia de Supervisor.

3 En la pestaña **Configurar**, seleccione **Supervisor > General**.

4 Seleccione **Servicio de espacio de nombres**.

5 Active o desactive el conmutador **Estado** para habilitar la función.

   Aparecerá la página **Crear plantilla de espacio de nombres**.

6 En el panel **Configuración**, establezca las limitaciones de recursos para el espacio de nombres.

| Opción | Descripción |
| --- | --- |
| CPU | La cantidad de recursos de CPU que se reservarán para el espacio de nombres. |
| Memoria | La cantidad de memoria que se reservará para el espacio de nombres. |
| Almacenamiento | La cantidad total de espacio de almacenamiento que se reservará para el espacio de nombres. |
| Directiva de almacenamiento | Establezca la cantidad de almacenamiento dedicado individualmente a cada una de las directivas de almacenamiento que asoció al espacio de nombres. |
| Clases de máquina virtual | Seleccione las clases de máquina virtual. Utilice Ctrl para seleccionar varias. |
| Biblioteca de contenido | Seleccione la biblioteca de contenido TKR. |

7 Haga clic en **Siguiente**.

8 En el panel **Permisos**, agregue ingenieros y grupos de desarrollo y operaciones para permitirles utilizar la plantilla con la que pueden crear espacios de nombres.

   a Seleccione el origen de identidad vsphere.local (debe utilizar vSphere SSO)

   b Seleccionar un usuario o un grupo

   c Seleccione la función de propietario

9 En el panel **Revisar y confirmar**, se muestran las propiedades que configura.

   Revise las propiedades y haga clic en **Listo**.

10 Compruebe que el servicio de espacio de nombres de vSphere esté activo.

Se configuró una plantilla de espacio de nombres de vSphere y tiene el estado activo. Los usuarios o grupos del espacio de nombres de vSphere asignados a la función Propietario pueden utilizar la plantilla para crear espacios de nombres de vSphere mediante el comando kubectl `kubectl create namespace <NAME>`.

# Quitar un espacio de nombres de vSphere

Puede quitar un espacio de nombres de vSphere de Supervisor. Antes de hacerlo, debe eliminar todos los clústeres de Servicio TKG que se aprovisionen allí.

## Requisito previo: eliminar los clústeres de Servicio TKG en espacio de nombres de vSphere

Antes de eliminar un espacio de nombres de vSphere, debe eliminar todos los clústeres de TKG 2.0 que se aprovisionen allí.

**Nota** También debe eliminar los pods de vSphere que estén implementados en el espacio de nombres de vSphere.

Utilice kubectl o la CLI de Tanzu para eliminar un clúster de TKG. Consulte Eliminar un clúster de TKG con Kubectl o la CLI de Tanzu.

**Nota** No intente eliminar un clúster de TKG mediante vSphere Client ni la CLI de vCenter Server.

## Quitar los espacio de nombres de vSphere

Para preparar el aprovisionamiento de clústeres de TKG 2.0, complete esta tarea para comprobar que el espacio de nombres de vSphere esté configurado correctamente.

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione Administración de cargas de trabajo.

3 Seleccione la pestaña **Espacio de nombres**.

Se mostrará cada espacio de nombres de vSphere creado en Supervisor.

4 Seleccione el espacio de nombres de vSphere que desee quitar.

5 Seleccione **Quitar**.

El sistema elimina el espacio de nombres de vSphere. El proceso puede tardar un tiempo en completarse. Utilice el panel Tareas para seguir el progreso.

# Aprovisionar clústeres del servicio de TKG

7

Con el servicio de TKG, puede aprovisionar dos tipos de clústeres de cargas de trabajo: clústeres de Tanzu Kubernetes y clústeres basados en ClusterClass.

Lea los siguientes temas a continuación:

- Acerca del aprovisionamiento de clústeres de TKG
- Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl
- Flujo de trabajo para aprovisionar clústeres de TKG mediante la CLI de Tanzu
- Probar el aprovisionamiento de clústeres de TKG mediante Kubectl
- Eliminar un clúster de TKG con Kubectl o la CLI de Tanzu
- Usar la API v1beta1 del clúster
- Usar la API v1alpha3 del clúster de Tanzu Kubernetes

## Acerca del aprovisionamiento de clústeres de TKG

El servicio TKG proporciona dos API y admite clientes para aprovisionar clústeres de TKG y administrar su ciclo de vida.

### Aprovisionamiento de clústeres de TKG

El diagrama muestra el flujo de trabajo para aprovisionar clústeres de TKG en Supervisor.

Figura 7-1. Aprovisionamiento de clústeres de TKG en Supervisor

Supervisor
TKG
Clúster de TKG
CP
Trabajadores
Estado del clúster declarado
Operador del clúster de TKG
vSphere
vCenter Server
ESXi
NSX
vSAN

## Tipos de clústeres de TKG

Existen dos tipos de clústeres de cargas de trabajo de Kubernetes que se pueden aprovisionar en la infraestructura de vSphere IaaS control plane en la que opera el Supervisor como clúster de administración y se aloja la API del clúster de Kubernetes (CAPI). Cada tipo se basa en la ClusterClass. Consulte las Notas de la versión de TKR para ver las versiones compatibles. Consulte también Uso de versiones de Kubernetes con clústeres de Servicio TKG.

**TanzuKubernetesCluster con el clúster de CAPI que hace referencia a la ClusterClass predeterminada denominada tanzukubernetescluster**

Firma de clúster:

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
```

Este tipo de clúster de carga de trabajo es un clúster de Tanzu Kubernetes compilado con la API del clúster (Cluster API, CAPI) que hace referencia a una ClusterClass predeterminada denominada `tanzukubernetescluster`. El tipo de clúster es **TanzuKubernetesCluster** y la API de aprovisionamiento es `v1alpha3`. Debido a que se trata de una abstracción encima de un clúster CAPI, la referencia a la clase de clúster de back-end no se especifica en el manifiesto del clúster. El sistema administra la referencia.

Con este tipo de clúster de carga de trabajo, el objeto TanzuKubernetesCluster está en la primera línea y funciona como capa de abstracción. No hay ningún cambio en el flujo de trabajo para aprovisionar este tipo de clúster que lo diferencie del aprovisionamiento de un clúster de TKGS en la versión 7 de vSphere IaaS control plane.

**Clúster de CAPI que hace referencia a la ClusterClass predeterminada denominada tanzukubernetescluster**

Firma de clúster:

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
```

Este tipo de clúster de carga de trabajo es un clúster de CAPI puro compilado con la API del clúster que hace referencia a una ClusterClass predeterminada denominada `tanzukubernetescluster`. El tipo de clúster es **Clúster** y la API de aprovisionamiento es `v1beta1`.

Con este tipo de clúster de carga de trabajo, la API del clúster de CAPI está en la primera línea; no hay ninguna capa de abstracción de TKC. El sistema proporciona un controlador para gestionar la infraestructura, por lo que no es necesario crear objetos relacionados con ClusterClass. Las variables se exponen para que pueda personalizar el clúster. Los campos de la especificación del clúster son diferentes de los campos de la especificación de TKC, pero el flujo de trabajo de aprovisionamiento es el mismo.

## API de aprovisionamiento de clústeres de TKG

TKG en Supervisor de vSphere 8 proporciona dos API para administrar el ciclo de vida de los clústeres de TKG: `v1alpha3` y `v1beta1`. Ambas API son de naturaleza declarativa, igual que la API de Kubernetes. Con el aprovisionamiento del clúster declarativo, se especifica el estado deseado del clúster de TKG: número de nodos, el almacenamiento disponible, los tamaños de máquina virtual y la versión de software de Kubernetes. TKG realiza el trabajo para aprovisionar y mantener un clúster que coincida con el estado declarado.

Si va a actualizar un clúster existente de Tanzu Kubernetes a TKG en Supervisor de vSphere 8, ese clúster debe utilizar la API `v1alpha2` antes de comenzar el proceso de actualización. Consulte la documentación de actualización para obtener más información: #unique_51.

| API | Variante | Versión de vCenter | Descripción |
| --- | --- | --- | --- |
| v1beta1 | Clúster | vCenter 8+ | Nueva API para administrar el ciclo de vida de un clúster basado en una clase de clúster. |
| v1alpha3 | TanzuKubernetesCluster | vCenter 8+ | Continuación de la API v1alpha2. Todas las funciones compatibles con la API v1alpha2 son compatibles con la API v1alpha3. Nuevas funciones añadidas. |

| API | Variante | Versión de vCenter | Descripción |
|---|---|---|---|
| v1alpha2 | TanzuKubernetesCluster | vCenter 7 U3 | API heredada para aprovisionar clústeres de Tanzu Kubernetes en supervisor de vCenter 7 U3 y para actualizar clústeres al supervisor de vCenter 8. Al actualizar o en vSphere 8, la API v1alpha2 se convierte automáticamente en la API v1alpha3. |
| v1alpha1 | TanzuKubernetesCluster | vCenter 7 U1, U2 | API obsoleta para aprovisionar clústeres de Tanzu Kubernetes en la primera generación del supervisor de vCenter 7. |

## Clientes de aprovisionamiento de clústeres de TKG

TKG en Supervisor de vSphere 8 admite diferentes flujos de trabajo de clientes para aprovisionar clústeres de TKG:

- Kubectl + YAML para el aprovisionamiento de clústeres declarativos de tipo Kubernetes. Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.
- CLI de Tanzu para el aprovisionamiento interactivo de clústeres de línea de comandos. Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante la CLI de Tanzu.
- Tanzu Mission Control para el aprovisionamiento de clústeres basado en web. Consulte Registrar Tanzu Mission Control alojado con Supervisor.

# Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl

Siga este flujo de trabajo para aprovisionar un clúster de servicio TKG de forma declarativa mediante los comandos kubectl y una especificación de clúster definida en YAML.

Este flujo de trabajo admite el aprovisionamiento de un clúster de TKG de forma declarativa mediante kubectl y YAML.

### Requisitos previos

Compruebe o complete los siguientes requisitos previos antes de iniciar el flujo de trabajo de aprovisionamiento:

- Instale o actualice el entorno a la versión de Supervisor más reciente. Consulte Capítulo 2 Ejecución de clústeres de Servicio TKG.
- Cree o actualice una biblioteca de contenido con las versiones de Tanzu Kubernetes más recientes. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.
- Cree y configure un espacio de nombres de vSphere para alojar clústeres de TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

Procedimiento

1 Instale Herramientas de la CLI de Kubernetes para vSphere.

Instalar el Herramientas de la CLI de Kubernetes para vSphere.

2 Autentíquese con Supervisor mediante kubectl.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN --vsphere-
username USERNAME
```

**Nota** El FQDN de Supervisor solo se puede utilizar si está habilitado. Consulte la documentación de configuración de Supervisor para obtener detalles.

3 Compruebe que el inicio de sesión en Supervisor se haya realizado correctamente.

Debe aparecer un error similar al siguiente:

```
Logged in successfully.

You have access to the following contexts:
   192.197.2.65
   tkg2-cluster-namespace
```

Donde `192.197.2.65` es el contexto de Supervisor y `tkg2-cluster-namespace` es el contexto para el espacio de nombres de vSphere donde planea aprovisionar el clúster de TKG.

4 Compruebe que el destino espacio de nombres de vSphere sea el contexto actual.

```
kubectl config get-contexts
```

```
CURRENT   NAME                    CLUSTER
AUTHINFO                                        NAMESPACE
          192.197.2.65            192.197.2.65
wcp:10.197.154.65:administrator@vsphere.local
*         tkg2-cluster-namespace  10.197.154.65
wcp:10.197.154.65:administrator@vsphere.local   tkg2-cluster-namespace
```

Si el espacio de nombres de vSphere de destino no es el contexto actual, cambie a él.

```
kubectl config use-context tkg2-cluster-namespace
```

5 Enumere las clases de máquina virtual que están disponibles en espacio de nombres de vSphere.

```
kubectl get virtualmachineclass
```

Solo puede utilizar las clases de máquina virtual que están enlazadas al espacio de nombres de destino. Si no ve ninguna clase de máquina virtual, compruebe que las clases de máquina virtual predeterminadas se hayan asociado con espacio de nombres de vSphere. Consulte también Corregir errores de clase de máquina virtual.

6 Obtiene las clases de almacenamiento de volumen persistente disponibles.

```
kubectl describe namespace VSPHERE-NAMESPACE-NAME
```

El comando devuelve detalles sobre el espacio de nombres, incluida la clase de almacenamiento con el formato `tkg2-storage-policy.storageclass.storage.k8s.io/requests.storage`. El primer token de la cadena es el nombre de la clase de almacenamiento, en este ejemplo, `tkg2-storage-policy`. El comando `kubectl describe storageclasses` también devuelve las clases de almacenamiento disponibles, pero requiere permisos de administrador de vSphere.

7 Lista de versiones de Tanzu Kubernetes disponibles:

Puede utilizar cualquiera de los siguientes comandos para realizar esta operación:

```
kubectl get tkr
```

```
kubectl get tanzukubernetesreleases
```

Este comando devuelve las TKR que están disponibles en este espacio de nombres de vSphere y muestra su compatibilidad con la instancia de Supervisor en la que se van a implementar. Solo puede utilizar esas versiones que devuelve este comando. Si no ve ninguna versión ni las versiones que desea, asegúrese de haber hecho lo siguiente: a) creó una biblioteca de contenido de TKR; b) sincronizó la biblioteca de contenido con los archivos OVA deseados; y c) asoció la biblioteca de contenido con el espacio de nombres de vSphere donde está aprovisionando el clúster de TKG.

8 Disetique el archivo YAML para aprovisionar el clúster de TKG.

a Determine el tipo de clúster que creará y revise su API y sus funciones:

- Clúster de Tanzu Kubernetes: Usar la API v1alpha3 del clúster de Tanzu Kubernetes
- Clúster: Usar la API v1beta1 del clúster

b Comience con una de las YAML de ejemplo para aprovisionar el clúster de.

Por ejemplo:

- Ejemplo de v1alpha3: clúster de Tanzu Kubernetes predeterminado
- Ejemplo de v1beta1: clúster predeterminado

c Guarde el archivo YAML como `tkg2-cluster-1.yaml` o un formato similar.

d Rellene el archivo YAML con la información que recopiló a partir de los resultados de los comandos anteriores, entre los que se incluyen:

- El nombre del clúster, como `tkg2-cluster-1`
- El espacio de nombres de vSphere de destino
- Las clases de máquina virtual enlazadas, como `guaranteed-medium`
- Clase de almacenamiento para nodos de clúster y volúmenes persistentes
- El número de nodos de trabajo y plano de control (réplicas)
- La versión de Tanzu Kubernetes especificada por la cadena NAME de TKR, como `v1.25.7+vmware.3-fips.1-tkg.1`

e Personalice el archivo YAML del clúster de TGK según sea necesario.

- Agregar volúmenes separados para componentes de renovación alta, como `containerd`
- Especificar una clase de almacenamiento persistente predeterminada para los nodos del clúster y los volúmenes persistentes
- Personalizar las redes de clústeres, incluidos los CIDR de CNI, pod y servicio

f Utilice un comprobador de sintaxis de YAML y compruebe que el archivo YAML sea válido.

El resultado de este paso es un YAML válido para aprovisionar el clúster de TKG.

9 Ejecute el siguiente comando para aprovisionar el clúster de TKG.

```
kubectl apply -f tkg2-cluster-1.yaml
```

Resultado esperado:

```
tanzukubernetescluster.run.tanzu.vmware.com/tkg2-cluster-1 created
```

10 Supervise el aprovisionamiento del clúster de TKG.

```
kubectl get tanzukubernetesclusters
```

```
kubectl get tkc
```

O bien, si crea un clúster de mediante la API v1beta1:

```
kubectl get cluster
```

Inicialmente, el estado LISTO es False mientras se aprovisiona el clúster. Después de unos minutos, debe ser True.

```
NAME               CONTROL PLANE   WORKER   TKR NAME                           AGE
READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkg2-cluster-1    3               6        v1.25.7+vmware.3-fips.1-tkg.1       49m
True    True
```

Ejecute comandos adicionales para ver los detalles del clúster.

```
kubectl get
tanzukubernetescluster,cluster,virtualmachinesetresourcepolicy,virtualmachineservice,kubead
mcontrolplane,machinedeployment,machine,virtualmachine
```

```
kubectl describe tanzukubernetescluster tkg2-cluster-1
```

11 Supervise la implementación de nodos del clúster mediante vSphere Client.

En el inventario de vSphere para **hosts y clústeres**, debería ver las máquinas virtuales de nodos de clústeres que se están implementando en el espacio de nombres de vSphere de destino.

12 Una vez que todos los nodos del clúster de TKG estén en estado LISTO, inicie sesión en el clúster mediante el complemento de vSphere para kubectl.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN \
--vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

Por ejemplo:

```
kubectl vsphere login --server=192.197.2.65 --vsphere-username user@vsphere.local \
--tanzu-kubernetes-cluster-name tkg2-cluster-1 --tanzu-kubernetes-cluster-namespace tkg2-
cluster-namespace
```

**Nota** El comando de inicio de sesión solo se ejecutará correctamente una vez que los nodos del plano de control se estén ejecutando y se haya iniciado el complemento del servicio de autenticación. Si los nodos de trabajo están en proceso de creación. Es posible que se detecte el inicio de sesión. Se recomienda iniciar sesión una vez que todos los nodos del clúster estén en estado LISTO.

13 Cambie el contexto al clúster de TKG para que sea el contexto actual.

Al iniciar sesión correctamente en el clúster de TKG, debería aparecer un mensaje similar al siguiente.

```
Logged in successfully.
```

```
You have access to the following contexts:
   192.197.2.65
   tkg2-cluster-namespace
   tkg2-cluster-1
```

Donde `192.197.2.65` es el contexto de Supervisor, `tkg2-cluster-namespace` es el contexto de espacio de nombres de vSphere y `tkg2-cluster-1` es el contexto del clúster de TKG.

Cambie al contexto del clúster de TKG.

```
kubect config use-context tkg2-cluster-1
```

14 Compruebe los recursos del clúster de TKG.

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl get pods -A
```

```
kubectl cluster-info
```

```
kubectl api-resources
```

15 Ejecute el clúster de TKG mediante la implementación de un pod de prueba y compruebe que funciona según lo esperado.

Consulte Probar el aprovisionamiento de clústeres de TKG mediante Kubectl.

# Flujo de trabajo para aprovisionar clústeres de TKG mediante la CLI de Tanzu

Siga este flujo de trabajo para aprovisionar un clúster de TKG v1beta1 mediante la CLI de Tanzu.

## Requisitos previos

Compruebe o complete los siguientes requisitos previos antes de iniciar el flujo de trabajo de aprovisionamiento:

- Instale o actualice el entorno a la versión de Supervisor más reciente. Consulte Capítulo 2 Ejecución de clústeres de Servicio TKG.
- Cree o actualice una biblioteca de contenido con las versiones de Tanzu Kubernetes más recientes. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.
- Cree y configure un espacio de nombres de vSphere para alojar clústeres de TKG 2.0. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

# Aprovisionar un clúster de TKG predeterminado

Complete estos pasos para aprovisionar un clúster v1beta1 predeterminado mediante la CLI de Tanzu. Para obtener más instrucciones o solucionar problemas, consulte Crear clústeres de carga de trabajo en la documentación de TKG independiente.

1 Instale la CLI de Tanzu.

   Consulte Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG.

2 Conéctese a Supervisor mediante la CLI de Tanzu.

   - Conectarse a Supervisor mediante la CLI de Tanzu y la autenticación de vCenter SSO o
   - Conectarse a Supervisor mediante la CLI de Tanzu y un IDP externo

3 Enumere las TKr disponibles.

```
tanzu kubernetes-release get
```

4 Cree un manifiesto de clúster con las configuraciones deseadas.

   Con TKG en Supervisor de vSphere 8, puede utilizar una especificación de objeto de estilo Kubernetes con la CLI de Tanzu para crear un clúster basado en ClusterClass.

   a Comenzar con Ejemplo de v1beta1: clúster predeterminado

   b Rellene `spec.clusterNetwork` con los `cidrBlocks` necesarios.

   c Rellene `spec.topology` con los valores esperados que aparecen en la tabla.

      - Cadena de NAME TKR, como `v1.26.13---vmware.1-fips.1-tkg.3`
      - Número de nodos del plano de control, como `3`
      - Nombre de cada grupo de nodos de trabajo, como `node-pool-1`
      - Número de nodos de trabajo, como `3`
      - Clase de máquina virtual, como `guaranteed-medium`
      - Clase de almacenamiento, como `tkg2-storage-policy`

5 Guarde el manifiesto del clúster como `cluster-default.yaml` y valide mediante un comprobador de YAML.

6 Cree el clúster de TKG.

```
tanzu cluster create -f cluster-default.yaml
```

   **Nota** Anexe `-v 8` para obtener una salida detallada.

7 Compruebe que se haya creado el clúster de TKG.

```
Workload cluster 'cluster-default' created
```

8 Después de crear el clúster, ejecute el siguiente comando para comprobar el estado del clúster.

```
tanzu cluster get cluster-default
```

9 Incluya el clúster.

```
tanzu cluster list
```

10 Compruebe los nodos del clúster.

```
tanzu cluster machinehealthcheck node get cluster-default
```

```
tanzu cluster machinehealthcheck control-plane get cluster-default
```

11 Obtenga el contexto de configuración para el clúster de TKG.

```
tanzu cluster kubeconfig get cluster-default -n tkg2-cluster-ns
```

12 Acceda al clúster.

```
kubectl config use-context tanzu-cli-cluster-default@cluster-default
```

13 Ejecute el clúster de TKG 2.0 mediante la implementación de un pod de prueba y compruebe que funciona según lo esperado.

Consulte Probar el aprovisionamiento de clústeres de TKG mediante Kubectl.

## Aprovisionar un clúster de TKG personalizado en Supervisor

Para aprovisionar un clúster de v1beta1 personalizado, como Ejemplo de v1beta1: clúster con CNI de Calico, puede incluir todas las especificaciones en un único YAML, como se indica en el ejemplo, cambiar ciertos valores para que coincidan con el entorno y ejecutar `kubectl apply -f cluster-calico.yaml`, por ejemplo.

Para aprovisionar el mismo clúster de v1beta1 personalizado mediante la CLI de Tanzu, los objetos de configuración `CalicoConfig` y `ClusterBootstrap` deben existir antes de crear el clúster.

Para aprovisionar el clúster con CNI de Calico:

1 Cree el YAML para los objetos de configuración `CalicoConfig` y `ClusterBootstrap`, con el nombre de clúster y el espacio de nombres deseados en cada uno.

2 Ejecute `kubectl apply -f` para cada uno de los tres objetos de configuración, o póngalos en un único YAML y ejecute `kubectl apply -f`.

3 Cree la especificación del clúster `cluster-calico.yaml` con el nombre y el espacio de nombres que coincidan con los de los objetos de configuración y cualquier otro parámetro deseado.

4 Cree el clúster.

```
tanzu cluster create -f cluster-calico.yaml
```

# Probar el aprovisionamiento de clústeres de TKG mediante Kubectl

Después de aprovisionar un clúster de TKG, se recomienda implementar una carga de trabajo de prueba y validar la funcionalidad del clúster.

Implemente una aplicación de prueba para verificar que el clúster de TKG esté activo y en ejecución.

#### Requisitos previos

- Aprovisione un clúster de TKG.
- Conéctese al clúster de TKG.

#### Procedimiento

1 Aprovisione un clúster de TKG.

Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

2 Inicie sesión en Supervisor mediante kubectl.

```
kubectl vsphere login --server=<IP or FQDN> --vsphere-username <USERNAME>
```

3 Cambie el contexto de configuración a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster de TKGS.

```
kubectl config use-context VSPHERE-NAMESPACE
```

4 Inicie sesión en el clúster de TKG de destino.

```
kubectl vsphere login --server=<IP or FQDN> --vsphere-username <USERNAME> \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

5 Cree el archivo `ping-pod.yaml` con el siguiente contenido.

```
apiVersion: v1
kind: Pod
metadata:
  name: ping-pod
  namespace: default
spec:
  containers:
  - image: busybox:1.34
    name: busybox
    command: ["ping", "-c"]
```

```
    args: ["1", "8.8.8.8"]
  imagePullSecrets:
  - name: regcred
  restartPolicy: Never
```

6 Cree la credencial del registro de `regcred`.

La imagen de contenedor utilizada para este escenario (BusyBox) se extrae del registro público de Docker Hub, lo que puede restringir las extracciones de imágenes. Si eso sucede, necesitará una cuenta de Docker Hub y un secreto de extracción de imágenes ("regcred") a los que se haga referencia en la especificación del pod. Para crear este secreto, consulte Crear secreto de credencial de registro privado.

7 Configure la seguridad del pod según sea necesario.

Si utiliza Tanzu Kubernetes 1.24 o una versión anterior, continúe con el siguiente paso y cree el pod.

Si utiliza la versión Tanzu Kubernetes 1.25, las Configurar PSA para TKR 1.25 y versiones posteriores estarán habilitadas. Puede continuar con el siguiente paso y crear el pod. Sin embargo, tenga en cuenta que recibirá una advertencia sobre infracciones de seguridad del pod, que puede ignorar.

```
Warning: would violate PodSecurity "restricted:latest": allowPrivilegeEscalation != false
(container "busybox" must set securityContext.allowPrivilegeEscalation=false),
unrestricted capabilities (container "busybox" must set
securityContext.capabilities.drop=["ALL"]),
runAsNonRoot != true (pod or container "busybox" must set
securityContext.runAsNonRoot=true),
seccompProfile (pod or container "busybox" must set securityContext.seccompProfile.type to
"RuntimeDefault" or "Localhost")
```

Si utiliza Tanzu Kubernetes 1.26 o una versión posterior, se aplicarán las Configurar PSA para TKR 1.25 y versiones posteriores. Si intenta crear el pod como se muestra en el próximo paso, se producirá el siguiente error.

```
Error from server (Forbidden): error when creating "ping-pod.yaml": pods "ping-pod" is
forbidden:
violates PodSecurity "restricted:latest": allowPrivilegeEscalation != false
(container "busybox" must set securityContext.allowPrivilegeEscalation=false),
unrestricted capabilities (container "busybox" must set
securityContext.capabilities.drop=["ALL"]),
runAsNonRoot != true (pod or container "busybox" must set
securityContext.runAsNonRoot=true),
seccompProfile (pod or container "busybox" must set securityContext.seccompProfile.type to
"RuntimeDefault" or "Localhost")
```

Para solucionar este problema, ejecute el siguiente comando en el espacio de nombres `default` donde se crea el pod. Tenga en cuenta que, al realizar esta acción, se eliminan las Configurar PSA para TKR 1.25 y versiones posteriores en el espacio de nombres.

```
kubectl label --overwrite ns default pod-security.kubernetes.io/enforce=privileged
```

Si lo prefiere, puede aplicar securityContext directamente al pod, por ejemplo:

```
...
spec:
  containers:
  - image: busybox:1.34
    name: busybox
    command: ["ping", "-c"]
    args: ["1", "8.8.8.8"]
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: ["ALL"]
      runAsNonRoot: true
      runAsUser: 1000
      seccompProfile:
        type: "RuntimeDefault"
...
```

8 Aplique el archivo YAML.

```
kubectl apply -f ping-pod.yaml
```

Resultado esperado:

```
pod/ping-pod created
```

9 Compruebe que el pod se haya completado correctamente.

```
kubectl get pods -n default
```

```
NAME       READY   STATUS      RESTARTS   AGE
ping-pod   0/1     Completed   0          13s
```

10 Verifique que el pod haya hecho ping al servidor DNS.

```
kubectl logs ping-pod -f
```

Resultado esperado:

```
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=106 time=33.352 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 33.352/33.352/33.352 ms
```

11 Elimine el pod.

```
kubectl delete -f ping-pod.yaml
```

12 Compruebe que el pod se haya eliminado.

```
kubectl get pods
```

# Eliminar un clúster de TKG con Kubectl o la CLI de Tanzu

Utilice `kubectl` o la CLI de Tanzu para eliminar un clúster de TKG.

Cuando se elimina un clúster de Tanzu Kubernetes mediante `kubectl` o la CLI de Tanzu, la recopilación de elementos no utilizados de Kubernetes garantiza que se eliminen todos los recursos dependientes.

**Nota** No intente eliminar un clúster mediante vSphere Client ni la CLI de vCenter Server.

### Procedimiento

1 Realice la autenticación con Supervisor.

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisionó el TKG que desea eliminar.

```
kubectl config use-context CLUSTER-NAMESPACE
```

3 Enumere los clústeres de TKG en el espacio de nombres.

```
kubectl get clusters
```

```
tanzu cluster list
```

4 Elimine el clúster de TKG con la siguiente sintaxis.

Clúster v1alpha3 con kubectl:

```
kubectl delete tanzukubernetescluster --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

Clúster v1beta1 con kubectl:

```
kubectl delete cluster --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

Clúster v1alpha3 o v1beta1 mediante la CLI de Tanzu:

```
tanzu cluster delete --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

Resultado de ejemplo:

```
tanzukubernetescluster.run.tanzu.vmware.com "tkg-cluster-1" deleted
```

5 Compruebe que el clúster se haya eliminado.

kubectl:

```
kubectl get clusters
```

CLI de Tanzu:

```
tanzu cluster list
```

6 Elimine el contexto del clúster del archivo `kubeconfig`.

```
kubectl config delete-context CONTEXT
```

Por ejemplo:

```
kubectl config get-contexts
CURRENT   NAME            CLUSTER         AUTHINFO
NAMESPACE
          192.0.2.1       192.0.2.1       wcp:192.0.2.1:administrator@vsphere.local
          tkg-cluster-1   192.0.2.6       wcp:192.0.2.6:administrator@vsphere.local
*         tkg-ns-1        192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkg-ns-1
```

```
kubectl config delete-context tkg-cluster-1
deleted context tkg-cluster-1 from $HOME/.kube/config
```

```
kubectl config get-contexts
CURRENT   NAME            CLUSTER        AUTHINFO
NAMESPACE
          192.0.2.1       192.0.2.1      wcp:192.0.2.1:administrator@vsphere.local
*         tkg-ns-1        192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkg-ns-1
```

# Usar la API v1beta1 del clúster

En esta sección, se proporciona contenido de referencia para aprovisionar un clúster mediante la API de v1beta1, incluidos ejemplos con diversas configuraciones y personalizaciones para satisfacer sus necesidades.

## API de clúster v1beta1

La API del clúster v1beta1 le permite aprovisionar un clúster en función de una definición ClusterClass predeterminada.

## API ClusterClass v1beta1

La API de clúster de Kubernetes es un conjunto de herramientas que admite para aprovisionar, actualizar y operar de forma declarativa los clústeres de Kubernetes. ClusterClass es una evolución de la API de clúster que permite definir plantillas para administrar el ciclo de vida de los conjuntos de clústeres. Servicio TKG admite ClusterClass mediante la API v1beta1.

Servicio TKG se incluye con una definición ClusterClass predeterminada denominada `tanzukubernetescluster`. ClusterClass de `tanzukubernetescluster` proporciona la plantilla para la creación de clústeres mediante la API v1beta. La opción ClusterClass de `tanzukubernetescluster` se encuentra disponible en todos los espacios de nombres de usuario. Para crear un clúster basado en esta instancia de ClusterClass, haga referencia a esta opción en la especificación del clúster. Consulte los ejemplos de v1beta para obtener instrucciones.

### tanzukubernetescluster de ClusterClass predeterminada

La ClusterClass `tanzukubernetescluster` predeterminada es inmutable. Puede actualizarse con cada versión del servicio TKG.

Para ver la variable ClusterClass `tanzukubernetescluster` predeterminada que se incluye con la instancia del servicio TKG, realice los siguientes pasos:

1 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=IP-or-FQDN --vsphere-username USER@vsphere.local
```

2 Cambie el contexto al espacio de nombres de vSphere en el que se aprovisiona un clúster de TKGS.

```
kubeclt config use-context VSPEHRE-NS
```

3 Obtenga la variable ClusterClass `tanzukubernetescluster` predeterminada.

```
kubectl get clusterclass tanzukubernetescluster -o yaml
```

4 Escriba el resultado de la variable ClusterClass predeterminada en un archivo denominado *tkc-dcc.yaml*.

```
kubectl get clusterclass tanzukubernetescluster -o yaml > tkc-dcc.yaml
```

### Variables ClusterClass para personalizar un clúster

Un clúster se personaliza en función de la ClusterClass `tanzukubernetescluster` mediante variables. Las variables se definen mediante pares nombre-valores. La sintaxis debe cumplir el esquema openAPIV3Schema.

Se requieren dos variables para aprovisionar un clúster mediante la API v1beta1:

- Clase de VM
- Clase de almacenamiento

Hay variables adicionales disponibles para personalizar un clúster, como las siguientes:

- Proxy
- Certificados TLS
- claves SSH

En las siguientes secciones se enumeran todas las variables que están disponibles con la ClusterClass`tanzukubernetescluster` predeterminada.

**Importante** Un nombre de clave válido solo debe constar de caracteres alfanuméricos, un guion (como `key-name`), un guion bajo (como `KEY_NAME`) o un punto (como `key.name`). No puede utilizar un espacio en un nombre de clave.

## clusterEncryptionConfigYaml

Utilice la variable `clusterEncryptionConfigYaml` para configurar el cifrado del clúster.

### clusterEncryptionConfigYaml

Cadena que es un archivo YAML que proporciona detalles de configuración de cifrado.

Puede configurar el cifrado de datos en la base de datos de etcd mediante el paquete de Configuración de cifrado kube-apiserver. Consulte Cifrar datos secretos en reposo en la documentación de Kubernetes.

```
...
    variables:
    #clusterEncryptionConfigYaml specifies the base64 encoded
    #EncryptionConfiguration YAML
    #the YAML contains a base64 encryption configuration for the cluster identity
    #the key is generated randomly
    - name: clusterEncryptionConfigYaml
      value: string which is name of the EncryptionConfiguration YAML
```

## controlPlaneCertificateRotation

Utilice la variable `controlPlaneCertificateRotation` para configurar el sistema para que rote los certificados TLS para los nodos del plano de control. Para ello, active una implementación de estos certificados antes de que caduquen. La rotación de certificados del plano de control estará disponible para todos los nodos de plano de control nuevos y existentes.

### controlPlaneCertificateRotation

Booleano para activar la función y el número de días antes de la caducidad a fin de rotar los certificados. Consulte Rotación automática de certificados mediante el proveedor de plano de control de Kubeadm para obtener más información.

```
...
    variables:
    - name: controlPlaneCertificateRotation
      value:
        activate: true
        daysBefore: 90
```

Donde:

- `activate` es un booleano para activar la función; el valor predeterminado es `true`
- `daysBefore` es la duración en el número de días antes de la fecha de caducidad; el valor predeterminado es `90` días. El mínimo es `7` días antes de que caduque

**Nota** Esta variable se actualiza para vSphere 8 Update 3 (Servicio TKG 3.0).

## controlPlaneVolumes

Utilice la variable `controlPlaneVolumes` para configurar volúmenes persistentes para los nodos del plano de control.

### controlPlaneVolumes

Matriz opcional de objetos; cada uno de ellos incluye `name`, `storageClass` y `mountPath`, cada uno de los cuales son cadenas, y un objeto `capacity` opcional que incluye una cadena `storage`.

```
...
    variables:
      #controlPlaneVolumes is an optional set of PVCs to create and
      #attach to each node
      - name: controlPlaneVolumes
        value:
          #name of the PVC to be used as the suffix (node.name)
          - name: NAME
            #mountPath is the directory where the volume device is mounted
            #takes the form /dir/path
            mountPath: /dir/path
            #storageClass is the storage class to use for the PVC
            storageClass: tkgs-storage-profile
            #capacity is the PVC storage capacity
```

```
              capacity:
                #storage sets the capacity for the disk volume
                #if not specified defaults to storageClass capacity
                storage: 4Gi
```

## defaultRegistrySecret

La variable `defaultRegistrySecret` configura el registro de contenedor predeterminado para el clúster.

**Nota** Esta variable está reservada para su uso con el registro de Harbor integrado.

### defaultRegistrySecret

Objeto que incluye una clave pública, un nombre de certificado y un espacio de nombres para el registro de contenedor predeterminado.

Si habilita el registro de Harbor integrado en Supervisor, la variable `defaultRegistrySecret` especifica el certificado del servicio de registro en el que confía el clúster. El secreto de certificado se etiqueta con `managed-by: vmware-vRegistry`. Al crear un clúster, se inserta un certificado defaultRegistry en la variable `defaultRegistrySecret`. Después de crear el clúster, puede administrar la rotación de certificados o cualquier actualización mediante la actualización manual de la variable.

```
...
    variables:
    - name: defaultRegistrySecret
      value:
        #data holds the base64 encoded data.ca\.crt content
        #data.ca\.crt is already encoded, so raw cert data is encoded twice
        data: LS0tLS1CRUdJTiBDRVJU...S0tRU5EIENFUlRJRklDQVRFL
        #name specifies the name of the registry cert secret
        name: harbor-ca-key-pair
        #namespace specifies the ns of the registry cert secret
        namespace: svc-harbor-domain-c9
```

## defaultStorageClass

Utilice la variable `defaultStorageClass` para configurar una clase de almacenamiento predeterminada para el clúster.

### defaultStorageClass

Cadena que identifica qué clase de almacenamiento se debe utilizar como clase de almacenamiento predeterminada, a menudo la requieren ciertas aplicaciones, como gráficos de Helm y paquetes de Tanzu.

```
...
    variables:
```

```
    - name: defaultStorageClass
      value: tkg2-storage-profile
```

## extensionCert

Utilice la variable `extensionCert` para configurar un certificado TLS.

### extensionCert

Objeto que contiene un objeto de `contentSecret` que contiene cadenas `name` y `key`. `contentSecret` hace referencia a un objeto secreto de Kubernetes que se creó para un certificado TLS.

```
...
    variables:
    #extensionCert specifies the cert and key for Extensions Controller
    #self-signed issuer and certificates must be created in advance
    - name: extensionCert
      value:
        contentSecret:
          #name specifies the name of secret
          name: string
          #key specifies the content of tls\.crt in the secret's data map
          key: string
```

## kubeAPIServerFQDNs

Utilice la variable `kubeAPIServerFQDNs` para configurar un clúster con un FQDN.

### kubeAPIServerFQDNs

Matriz de uno o varios nombres de dominio completos (FQDN).

El certificado de API de Kubernetes que se genera incluirá cada uno de los FQDN que haya especificado en la variable `kubeAPIServerFQDNs`. El sistema rellenará kubeconfig con el primer FQDN de la lista y asume que se puede resolver. Si desea utilizar un FQDN diferente de la lista, puede editar manualmente el archivo kubeconfig generado con el FQDN deseado de la lista de variables.

Consulte Ejemplo de v1beta1: clúster con FQDN para obtener detalles.

## nodePoolLabels

Utilice la variable `nodePoolLabels` para configurar etiquetas para los nodos de trabajo.

### nodePoolLabels

Matriz de uno o varios objetos; cada objeto contiene un par clave/valor y ambos son cadenas.

Las etiquetas le permiten organizar los objetos del sistema según sus requisitos y, de este modo, facilitar las consultas y la generación de informes. Consulte la documentación de etiquetas de Kubernetes para ver detalles sobre el uso.

## nodePoolTaints

Utilice la variable `nodePoolTaints` para aplicar manchas a los nodos de trabajo.

### nodePoolTaints

Matriz de objetos; cada objeto contiene un taint que se aplica a los nodos de trabajo.

Cada objeto taint incluye una `key` (cadena), un `value` (cadena) y un `effect` (cadena). El sistema rellena el campo `timeAdded` durante la creación o la actualización.

## nodePoolVolumes

Utilice la variable `nodePoolVolumes` para especificar volúmenes persistentes para los nodos del clúster.

### nodePoolVolumes

Matriz opcional de objetos; cada uno de ellos incluye `name`, `storageClass` y `mountPath`, cada uno de los cuales son cadenas, y un objeto `capacity` opcional que incluye una cadena `storage`.

```
...
    variables:
      #nodePoolVolumes is an optional set of PVCs to create and
      #attach to each node; use for high-churn components like containerd
      - name: nodePoolVolumes
        value: |
          #name of the PVC to be used as the suffix (node.name)
          - name: etcd
            #mountPath is the directory where the volume device is mounted
            #takes the form /dir/path
            mountPath: /var/lib/containerd
            #storageClass is the storage class to use for the PVC
            storageClass: tkgs-storage-profile
            #capacity is the PVC storage capacity
            capacity:
              #storage sets the capacity for the disk volume
              #if not specified defaults to storageClass capacity
              storage: 4Gi
```

## ntp

Utilice la variable `ntp` para configurar un servidor NTP para el clúster.

### ntp

Cadena que es el FQDN o la dirección IP de un servidor NTP.

La variable NTP especifica el nombre de dominio del servidor NTP como se muestra en el ejemplo. El servidor NTP se inserta en la variable Clúster al crear el clúster. Después de crear el clúster, puede administrar la rotación de nombres de servidor o cualquier actualización mediante la actualización manual de la variable del clúster.

```
...
    variables:
    - name: ntp
      value: time1.vmware.com
```

## podSecurityStandard

Utilice la variable `podSecurityStandard` para configurar la seguridad de los pods en todo el clúster.

**Nota** Esta variable es nueva para vSphere 8 Update 3 (Servicio TKG 3.0).

### podSecurityStandard

Con TKr v1.26 y versiones posteriores, de forma predeterminada, las restricciones de seguridad de pods (PSA) se aplican en el nivel del espacio de nombres mediante etiquetas de anotación. Consulte Configurar PSA para TKR 1.25 y versiones posteriores.

Si lo prefiere, puede utilizar la variable `podSecurityStandard` para configurar PSA en todo el clúster al aprovisionar o actualizar un clúster v1beta1.

La variable `podSecurityStandard` puede implementarse de la siguiente manera:

```
...
variables:
- name: podSecurityStandard
  value:
    deactivated: DEACTIVATED
    audit: AUDIT-PROFILE
    enforce: ENFORCE-PROFILE
    warn: WARN-PROFILE
    auditVersion: AUDIT-VERSION
    enforceVersion: ENFORCE-VERSION
    warnVersion: WARN-VERSION
    exemptions:
      namespaces: [EXEMPT-NS]
```

Donde:

- El valor DESACTIVADO es `false` (valor predeterminado) para aplicar PSA en todo el clúster y, de lo contrario, `true`.
- El valor *-PROFILE es el perfil de PSA de cada modo, que puede ser `"privileged"`, `"baseline"` o `"restricted"` (predeterminado).
- El valor *-VERSION es la versión de Kubernetes para cada modo, como `"v1.26"`. El valor `"latest"` es el predeterminado.

- El valor EXEMPT-NS es una lista separada por comas de espacios de nombres que se excluirán del control de PSA.

**Nota** Los espacios de nombres del sistema se excluyen de la seguridad de pods, incluidos kube-system, tkg-system y vmware-system-cloud-provider.

Si no implementa la variable `podSecurityStandard`, se conserva el comportamiento de PSA predeterminado. Si incluye la variable `podSecurityStandard` en la especificación del clúster, la configuración de la variable controlará, incluidos sus valores predeterminados, a menos que los anule.

En el siguiente ejemplo se muestran los valores predeterminados.

```
...
    variables:
      - name: podSecurityStandard
        value:
          enforce: "restricted"
          enforce-version: "latest"
```

En el siguiente ejemplo se proporcionan registros de auditoría y advertencias para identificar las cargas de trabajo que no siguen las prácticas recomendadas de protección de pods actuales, pero solo se aplica una directiva mínimamente restrictiva ("línea base") que impide la ampliación de privilegios conocidos.

```
...
    variables:
      - name: podSecurityStandard
        value:
          audit: "restricted"
          warn: "restricted"
          enforce: "baseline"
```

En el siguiente ejemplo se aplica una directiva restringida, excepto en un espacio de nombres específico.

```
...
    variables:
      - name: podSecurityStandard
        value:
          audit: "restricted"
          warn: "restricted"
          enforce: "restricted"
          exemptions:
            namesaces: ["privileged-workload-ns"]
```

El siguiente ejemplo restringe la aplicación a una versión de TKr en particular.

```
...
variables:
  - name: podSecurityStandard
    value:
      audit-version: "v1.26"
      warn-version: "v1.26"
      enforce-version: "v1.26"
```

Para ver más ejemplos, consulte los estándares de seguridad de pods en la documentación de Kubernetes.

## proxy

Utilice la variable `proxy` para configurar un servidor proxy para el clúster.

### proxy

Objeto con parámetros que hacen referencia a un servidor proxy para las conexiones de clústeres salientes.

Los parámetros requeridos del `proxy` son `httpProxy`, `httpsProxy` y `noProxy`. Los tres campos son obligatorios si incluye la variable `proxy` en la definición del clúster.

Los campos `httpProxy` y `httpsProxy` toman los valores de cadenas que hacen referencia al URI de un servidor proxy que está configurado para administrar las conexiones HTTP y HTTPS salientes desde el clúster de TKG. Puede conectarse al servidor proxy mediante HTTP. No se admiten conexiones HTTPS.

El campo `noProxy` es una matriz de cadenas. Obtenga los valores `noProxy` de la red de cargas de trabajo de Supervisor. Debe incluir en el campo `noProxy` las subredes de red de espacio de nombres, entrada y salida.

No es necesario incluir la subred de servicios en el campo `noProxy`. El clúster de TKG no interactúa con esta subred.

No es necesario incluir `clusterNetwork.services.cidrBlocks` y `clusterNetwork.pods.cidrBlocks` en el campo `noProxy`. Estos endpoints no son objeto de proxy automáticamente.

No es necesario incluir localhost y `127.0.0.1` en el campo `noProxy`. Estos endpoints no son objeto de proxy automáticamente.

```
...
    variables:
    #proxy specifies a proxy server to be used for the cluster
    #if omitted no proxy is configured
    - name: proxy
      value:
        #httpProxy is the proxy URI for HTTP connections
        #to endpoints outside the cluster
        httpProxy: http://<user>:<pwd>@<ip>:<port>
```

```
        #httpsProxy is the proxy URL for HTTPS connections
        #to endpoints outside the cluster
        httpsProxy: http://<user>:<pwd>@<ip>:<port>
        #noProxy is the list of destination domain names, domains,
        #IP addresses, and other network CIDRs to exclude from proxying
        #must include Supervisor Pod, Egress, Ingress CIDRs
        noProxy: [array of strings, comma-separated]
```

## storageClass

Utilice la variable `storageClass` para configurar una clase de almacenamiento para el clúster.

### storageClass

Cadena que es el nombre de un perfil de almacenamiento de vSphere que se ha asignado al espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG.

```
...
    variables:
    - name: storageClass
      value: tkgs-storage-profile
```

Utilice el siguiente comando para enumerar las clases de almacenamiento disponibles:

```
kubectl describe namespace VSPHERE-NAMESPACE-NAME
```

O bien, si tiene privilegios de administrador de vSphere:

```
kubectl describe storageclasses
```

## storageClasses

Utilice la variable `storageClasses` para configurar una matriz de clases de almacenamiento para el clúster.

### storageClasses

Matriz de una o varias cadenas; cada cadena es el nombre de un perfil de almacenamiento de vSphere que se ha asignado al espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG.

```
...
    variables:
    - name: storageClasses
      value: [tkg2-storage-profile, tkg2-storage-profile-latebinding]
```

Utilice el siguiente comando para enumerar las clases de almacenamiento disponibles:

```
kubectl describe namespace VSPHERE-NAMESPACE-NAME
```

O bien, si tiene privilegios de administrador de vSphere:

```
kubectl describe storageclasses
```

## TKR_DATA

Utilice la variable `TKR_DATA` para especificar la información de TKR.

### TKR_DATA

Objeto que utiliza para especificar la versión de TKR y otros detalles.

`version` es una cadena con el formato TKR NAME que utilizarán los nodos del clúster.

Solo las TKR que no tengan la etiqueta `legacy-tkr` son compatibles con TKG en el supervisor de vSphere 9. Consulte Uso de versiones de Kubernetes con clústeres de Servicio TKG.

El sistema operativo predeterminado es PhotonOS. Utilice anotaciones para especificar la Ejemplo de v1beta1: clúster con Ubuntu TKR.

## trust

Utilice la variable `trust` para especificar uno o varios certificados de CA de confianza para el clúster.

### trust

Objeto para agregar certificados TLS al clúster, ya sean certificados de entidad de certificación adicionales o finales.

El valor es `additionalTrustedCAs`, que es una matriz de cadenas. Por ejemplo:

```
#trust-example
    variables:
      - name: trust
        value:
          additionalTrustedCAs:
          - name: additional-ca-1
          - name: additional-ca-2
          - name: additional-ca-N
```

El valor de cada cadena es el nombre definido por el usuario para el campo de asignación de datos en el secreto de Kubernetes que contiene el certificado de CA en formato PEM con doble codificación base64. Por ejemplo:

```
#secret-example
apiVersion: v1
data:
  additional-ca-1: TFMwdExTMUNSGlSzZ3Jaa...VVNVWkpRMEMwdExTMHRDZz09
kind: Secret
metadata:
  name: cluster01-user-trusted-ca-secret
  namespace: tkgs-cluster-ns
type: Opaque
```

Un caso práctico común de la variable de confianza es integrar un clúster v1beta1 con un registro de contenedor privado. Consulte Integrar clústeres de Servicio TKG con un registro de contenedor privado

## usuario

Utilice la variable `user` para especificar las credenciales de usuario del clúster.

### usuario

Objeto que incluye un objeto passwordSecret, con cadenas de nombre y clave, y una cadena sshAuthorizedKey. Puede utilizar esta variable para agregar la clave SSH de un usuario a los nodos del clúster para el acceso SSH remoto.

La variable Usuario especifica las credenciales de inicio de sesión SSH, incluidas la contraseña y las claves autorizadas. El nombre de usuario de forma predeterminada es `vmware-system-user`. La contraseña debe estar cifrada con hash y almacenada en un secreto en el mismo espacio de nombres en el que se aprovisiona el clúster. El objeto `passwordSecret` hace referencia a este secreto. Por ejemplo, en Linux, puede generar un hash seguro mediante `mkpasswd --method=SHA-512 --rounds=4096`. Consulte Incluir usuarios y grupos para obtener información más detallada.

```
...
    variables:
    #user specifies an authorized user and credentials
    - name: user
      value:
        #passwordSecret is an object that contains a Kubernetes secret and key
        passwordSecret:
          #name specifies the secret name
          name: string
          #key specifies the key value pair in the secret's data map
          key: string
        sshAuthorizedKey: string that is the base64-encoded public key
```

## vmClass

Utilice la variable `vmClass` para configurar la clase de máquina virtual para los nodos del clúster.

### vmClass

Cadena obligatoria que se asigna al nombre de una clase de máquina virtual que está enlazada al espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG.

`vmClass` es el nombre de la VirtualMachineClass que describe la configuración del hardware virtual que se utilizará para los nodos del clúster. VirtualMachineClass controla la CPU y la memoria disponibles para el nodo, así como las solicitudes y los límites de esos recursos. Consulte Uso de clases de máquinas virtuales con clústeres de Servicio TKG.

Solo puede utilizar clases de máquinas virtuales que estén asociadas al espacio de nombres de vSphere en el que se aprovisiona el clúster de servicio TKG. Utilice el comando `kubectl get virtualmachineclass` para enumerar las clases enlazadas.

Puede definir la variable `vmClass` en diferentes ámbitos para poder utilizar diferentes clases de máquina virtual para los nodos de plano de control y los nodos de trabajo del grupo de nodos.

Por ejemplo, aquí una variable `vmClass` en línea `overrides` la variable `vmClass` principal para esta topología de machineDeployment específica.

```
...
    workers:
      machineDeployments:
      - class: tkg-worker
        name: compute
        replicas: 3
        variables:
          overrides:
          - name: vmClass
            value: guaranteed-large
```

## Ejemplo de v1beta1: clúster predeterminado

Consulte este ejemplo para aprovisionar un clúster v1beta1 con la configuración predeterminada.

### Ejemplo de v1beta1: clúster predeterminado

El siguiente ejemplo de YAML utiliza la API v1beta1 para crear un clúster predeterminado basado en la ClusterClass predeterminado.

Este ejemplo representa la configuración mínima necesaria para crear un clúster mediante la API v1beta1. El ejemplo se anota con descripciones de cada campo. Para obtener más información, consulte el código de origen.

Tenga en cuenta lo siguiente sobre este ejemplo:

- A diferencia de la API API v1alpha3 del clúster de Tanzu Kubernetes, la API v1beta1 requiere que especifique la `clusterNetwork`. No hay ninguna configuración de red predeterminada para el tipo de clúster.
- La ClusterClass predeterminada es `tanzukubernetescluster` que se documenta aquí: API de clúster v1beta1 .
- Puede personalizar el clúster mediante `variables`, cada una de las cuales es un par nombre-valor. Como mínimo, debe especificar las clases de máquina virtual y almacenamiento como variables, como se muestra en el ejemplo.

- Aunque técnicamente es opcional, el ejemplo también incluye una variable `defaultStorageClass`, ya que muchas cargas de trabajo, incluidos los paquetes de Tanzu y los gráficos de Helm, requieren que el clúster se aprovisione con una clase de almacenamiento predeterminada.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
#define the cluster
metadata:
  #user-defined name of the cluster; string
  name: cluster-default
  #kubernetes namespace for the cluster; string
  namespace: tkg-cluster-ns
#define the desired state of cluster
spec:
  #specify the cluster network; required, there is no default
  clusterNetwork:
    #network ranges from which service VIPs are allocated
    services:
      #ranges of network addresses; string array
      #CAUTION: must not overlap with Supervisor
      cidrBlocks: ["198.51.100.0/12"]
    #network ranges from which Pod networks are allocated
    pods:
      #ranges of network addresses; string array
      #CAUTION: must not overlap with Supervisor
      cidrBlocks: ["192.0.2.0/16"]
    #domain name for services; string
    serviceDomain: "cluster.local"
  #specify the topology for the cluster
  topology:
    #name of the ClusterClass object to derive the topology
    class: tanzukubernetescluster
    #kubernetes version of the cluster; format is TKR NAME
    version: v1.26.13---vmware.1-fips.1-tkg.3
    #describe the cluster control plane
    controlPlane:
      #number of control plane nodes
      #integer value 1 or 3
      #NOTE: Production clusters require 3 control plane nodes
      replicas: 3
    #describe the cluster worker nodes
    workers:
      #specifies parameters for a set of worker nodes in the topology
      machineDeployments:
        #node pool class used to create the set of worker nodes
        - class: node-pool
          #user-defined name of the node pool; string
          name: node-pool-1
          #number of worker nodes in this pool; integer 0 or more
          replicas: 3
    #customize the cluster
    variables:
      #virtual machine class type and size for cluster nodes
```

```
      - name: vmClass
        value: guaranteed-medium
      #persistent storage class for cluster nodes
      - name: storageClass
        value: tkg-storage-policy
      # default storageclass for control plane and worker node pools
      - name: defaultStorageClass
        value: tkg-storage-policy
```

# Ejemplo de v1beta1: clúster personalizado basado en la ClusterClass predeterminada

Consulte este ejemplo para aprovisionar un clúster v1beta1 con una configuración personalizada.

## Ejemplo de v1beta1: clúster personalizado basado en la ClusterClass predeterminada

El siguiente ejemplo de YAML demuestra cómo utilizar la API de v1beta1 para aprovisionar un clúster con varias opciones de configuración personalizadas mediante variables. Este ejemplo se basa en Ejemplo de v1beta1: clúster predeterminado.

En el ejemplo se utilizan variables de volúmenes persistentes en nodos de trabajo para componentes de renovación alta como containerd y kubelet. Además, la variable `vmClass` se declara dos veces. La variable is `vmClass` que se declara en `workers.machineDeployments` sobrescribe la variable `vmClass` que se declara globalmente para que los nodos de trabajo se aprovisionen con una clase de máquina virtual más grande.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-custom
  namespace: tkg-cluster-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.2.0.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.25.7---vmware.3-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 3
          variables:
            overrides:
            - name: vmClass
```

```
            value: guaranteed-xlarge
  variables:
    - name: vmClass
      value: guaranteed-medium
    - name: storageClass
      value: tkg-storage-profile
    - name: defaultStorageClass
      value: tkg-storage-profile
    - name: nodePoolVolumes
      value:
        - name: containerd
          capacity:
            storage: 50Gi
          mountPath: /var/lib/containerd
          storageClass: tkg-storage-profile
        - name: kubelet
          capacity:
            storage: 50Gi
          mountPath: /var/lib/kubelet
          storageClass: tkg-storage-profile
```

## Ejemplo de v1beta1: clúster con CNI de Calico

Consulte este ejemplo para aprovisionar un clúster v1beta1 con la CNI de Calico en lugar de la CNI de Antrea predeterminada. También puede consultar este ejemplo para obtener instrucciones sobre cómo personalizar uno o varios paquetes de TKR para un clúster.

### Ejemplo de v1beta1: clúster con CNI personalizada

El siguiente ejemplo de YAML demuestra cómo utilizar la API de v1beta1 para aprovisionar un clúster con una CNI personalizada. Este ejemplo se basa en Ejemplo de v1beta1: clúster predeterminado.

Como se define en `tanzukubernetescluster` ClusterClass, la CNI predeterminada es Antrea. La otra CNI admitida es Calico. Para cambiar la CNI de Antrea a Calico, sobrecargue la CNI predeterminada creando un recurso personalizado de `ClusterBootstrap` que haga referencia a un recurso personalizado de `CalicoConfig`.

El recurso personalizado de `ClusterBootstrap` incluye el bloque `spec.cni.refName` con el valor que proviene del TKR. (Consulte Paquetes de TKr para obtener instrucciones sobre cómo obtener el valor de este campo). El valor de `ClusterBootstrap` sobrescribe el valor predeterminado en ClusterClass y lo selecciona la API del clúster (CAPI) cuando se crea el clúster. El nombre del recurso personalizado `ClusterBootstrap` debe ser el mismo que el `Cluster`.

**Nota** El ejemplo se proporciona como un único archivo YAML, pero se puede separar en archivos individuales. Si hace esto, debe crearlos en orden: primero el recurso personalizado de `CalicoConfig` y, a continuación, el `ClusterBootstrap` y el clúster de `cluster-calico`.

```
---
apiVersion: cni.tanzu.vmware.com/v1alpha1
kind: CalicoConfig
metadata:
  name: cluster-calico
spec:
  calico:
    config:
      vethMTU: 0
---
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: ClusterBootstrap
metadata:
  annotations:
    tkg.tanzu.vmware.com/add-missing-fields-from-tkr: v1.23.8---vmware.2-tkg.2-zshippable
  name: cluster-calico
spec:
  cni:
    refName: calico.tanzu.vmware.com.3.22.1+vmware.1-tkg.2-zshippable
    valuesFrom:
      providerRef:
        apiGroup: cni.tanzu.vmware.com
        kind: CalicoConfig
        name: cluster-calico
---
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-calico
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.23.8---vmware.2-tkg.2-zshippable
    controlPlane:
      replicas: 3
    workers:
```

```
      machineDeployments:
        - class: node-pool
          name: node-pool-1
          replicas: 3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg2-storage-policy
```

# Ejemplo de v1beta1: clúster con Ubuntu TKR

Consulte este ejemplo para aprovisionar un clúster v1beta1 que utilice la edición Ubuntu de una versión de Tanzu Kubernetes.

## Ejemplo de v1beta1: clúster con Ubuntu TKR

El siguiente ejemplo de YAML demuestra cómo utilizar la API v1beta1 para aprovisionar un clúster que utiliza la edición Ubuntu del TKR especificado. Este ejemplo se basa en Ejemplo de v1beta1: clúster predeterminado.

De forma predeterminada, se utiliza PhotonOS para los nodos del clúster. Si la versión de TKR admite varias imágenes del sistema operativo, incluya la anotación `run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu` en la especificación del clúster para usar Ubuntu en lugar de Photon. Si utiliza la edición Ubuntu de la TKr, solo debe especificar la versión completa de la cadena e incluir la anotación del sistema operativo en la especificación del clúster. Para obtener más información sobre TKR, consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-ubuntu
  namespace: tkg-cluster-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.25.7---vmware.3-fips.1-tkg.1
    controlPlane:
      replicas: 3
      metadata:
        annotations:
          run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
    workers:
      machineDeployments:
        - class: node-pool
```

```
          name: node-pool-1
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
        - class: node-pool
          name: node-pool-2
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
        - class: node-pool
          name: node-pool-3
          replicas: 3
          metadata:
            annotations:
              run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg-storage-policy
```

# Ejemplo de v1beta1: clúster con FQDN

Consulte este ejemplo para aprovisionar un clúster de TKGS mediante uno o varios nombres de dominio completo (FQDN).

## Compatibilidad con FQDN

Puede utilizar la API v1beta1 para aprovisionar un clúster de TKG con un FQDN. La API v1beta1 del clúster incluye una variable denominada `kubeAPIServerFQDNs` que contiene al menos una cadena de FQDN que se generará en el certificado TLS para el servidor de API de Kubernetes.

Cuando se emite el comando `kubectl vsphere login` para un clúster con el FQDN configurado, el servicio de autenticación selecciona la primera entrada de FQDN de la lista y la agrega a kubeconfig como la opción preferida para interactuar con el clúster. Se supone que el primer FQDN de la lista se puede resolver. No se requieren cambios en el inicio de sesión del clúster.

El certificado de API de Kubernetes que se genere incluirá todos los FQDN que haya especificado en la variable `kubeAPIServerFQDNs`. El sistema no intentará utilizar ningún otro FQDN de la lista que no sea el primero. El sistema no intenta resolver el FQDN. Si desea utilizar un FQDN diferente de la lista, puede editar manualmente el archivo kubeconfig generado y agregar el FQDN deseado.

## Requisitos de FQDN

El uso de un FQDN es opcional. La funcionalidad no cambia si no utiliza uno. La funcionalidad que se describe aquí es específica de los clústeres de carga de trabajo de TKG. Para utilizar un FQDN con Supervisor, consulte ese tema en la documentación de Supervisor.

Cumpla los siguientes requisitos para aprovisionar un clúster de TKG con un FQDN.

- Use entornos vSphere 8.0 U2 P03 y versiones posteriores.
- Supervisor se actualizó a la versión de revisión más reciente.
- Solo se admiten clústeres de API v1beta1; no se admiten clústeres de API v1alpha3.
- DNS está configurado para resolver el FQDN seleccionado en una dirección IP válida.

**Importante** La función FQDN solo está disponible si se utiliza la API v1beta1 para aprovisionar un clúster de CAPI. No se puede aprovisionar un TKC mediante la API v1alpha3 con un FQDN.

## Ejemplo de FQDN

Utilice la API v1beta1 del clúster para crear un clúster con un FQDN.

El valor `spec.topology.variables.kupeAPIServerFQDNs` es una matriz de FQDN.

El sistema seleccionará el primer FQDN de la lista, que en este ejemplo es `demo.fqdn.com`.

```
#cluster-example-fqdn.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: tkg-cluster-fqdn
  namespace: tkg-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.52.100.0/12"]
    pods:
      cidrBlocks: ["192.101.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.26.5+vmware.2-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-01
          replicas: 3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkgs-storage-class
      - name: defaultStorageClass
        value: tkg-storage-class
      - name: kubeAPIServerFQDNs
        value:
          - demo.fqdn.com
          - explore.fqdn.com
```

## Verificación de FQDN

Complete el siguiente procedimiento para comprobar que el primer FQDN de la lista de variables esté incluido en el archivo `kubeconfig` y que todos los FQDN de la lista de variables estén en el certificado TLS del servidor de API de Kubernetes.

1 Inicie sesión en el clúster de TKG mediante Kubectl.

```
kubectl vsphere login --server=SVCP IP or FQDN --vsphere-username USERNAME --tanzu-
kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace VSPHERE-NS
```

2 Puede ver el FQDN en el archivo `kubeconfig`.

```
cat ~/.kube/config
```

3 Compruebe que la primera variable FQDN de la lista esté incluida en el archivo `kubeconfig`.

Por ejemplo:

```
apiVersion: v1
clusters:
- cluster:
    insecure-skip-tls-verify: false
    server: https://10.199.155.77:6443
  name: 10.199.155.77
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJ...DQWRLZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXc
    ...
    CkdiL1pua09rOVVjT3BwSStCTE9ZZDR0RGd2eHo...QUp0SUUKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
    server: https://demo.fqdn.com:6443
  name: demo.fqdn.com
```

4 Con vSphere Client, obtenga la dirección IP del clúster de TKGS en **Administración de cargas de trabajo >** ***Espacio de nombres*** **> Recursos informáticos > Clústeres de Tanzu Kubernetes > Dirección del plano de control**.

5 Realice una entrada de DNS manual en el archivo `/etc/hosts` local con la dirección IP y el FQDN.

Por ejemplo:

```
sudo vi /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu-client-vm
10.199.155.77 demo.fqdn.com
...
```

6 Utilice el comando `openssl s_client` para ver el certificado TLS.

```
echo | openssl s_client -servername hostname -connect FQDN:PORT 2>/dev/null | openssl x509
-text
```

`FQDN` es el primero en la lista de variables `kubeAPIServerFQDNs`.

Por ejemplo:

```
echo | openssl s_client -servername hostname -connect demo.fqdn.com:6443 2>/dev/null |
openssl x509 -text
```

7 En el campo `Subject Alternative Name`, deben estar todos los FQDN incluidos.

```
X509v3 Subject Alternative Name:
    DNS:demo.fqdn.com, DNS:explore.fqdn.com, DNS:kubernetes, DNS:kubernetes.default,
DNS:kubernetes.default.svc, DNS:kubernetes.def
```

Debido a que el certificado TLS para el servidor de API de Kubernetes incluye todos los FQDN de la lista de `kubeAPIServerFQDNs`, puede actualizar manualmente el archivo `kubeconfig` para utilizar el segundo (o tercer, etc.) FQDN en la lista y esto debería funcionar (suponiendo que se pueda resolver).

## Ejemplo de v1beta1: clúster entre zonas de vSphere

Consulte este ejemplo para aprovisionar un clúster v1beta1 en Supervisor implementado en tres Zonas de vSphere.

### Ejemplo de v1beta1: clúster entre zonas de vSphere

El siguiente ejemplo de YAML utiliza la API v1beta1 para aprovisionar un clúster en una topología de zona de vSphere. Este ejemplo se basa en Ejemplo de v1beta1: clúster predeterminado.

En este ejemplo se implementan varios grupos de nodos de trabajo. Cada grupo de nodos hace referencia a un dominio de errores que se asigna a una zona de vSphere. Para obtener más información sobre las zonas de vSphere, consulte *Instalar y configurar el plano de control de IaaS de vSphere*.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster-zoned
  namespace: tkg-cluster-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.25.7---vmware.3-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      #muliple node pools are used
      machineDeployments:
        - class: node-pool
```

```
          name: node-pool-1
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: vsphere-zone1
        - class: node-pool
          name: node-pool-2
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: vsphere-zone2
        - class: node-pool
          name: node-pool-3
          replicas: 3
          #failure domain the machines will be created in
          #maps to a vSphere Zone; name must match exactly
          failureDomain: vsphere-zone3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkg-storage-policy
```

## Ejemplo de v1beta1: clúster con red de pods enrutables

La API v1beta1 se puede utilizar para crear un clúster con una red de pods enrutables. Para ello, se reemplaza el clúster predeterminado con configuraciones personalizadas para `AntreaConfig` y `VSphereCPIConfig`.

### Acerca de las redes de pods enrutables mediante la API v1beta1

El siguiente ejemplo de YAML demuestra cómo utilizar la API v1beta1 para aprovisionar un clúster con una función RoutablePod de Antrea habilitada. Este ejemplo se basa en el ejemplo de v1beta1: clúster predeterminado.

Para habilitar la función RoutablePod, el clúster requiere `AntreaConfig` y `VSphereCPIConfig` con una configuración especial.

`AntreaConfig` debe establecer `trafficEncapMode: noEncap` y `noSNAT: true`.

`VSphereCPIConfig` debe establecer `antreaNSXPodRoutingEnabled: true`, `mode: vsphereParavirtualCPI` y

```
tlsCipherSuites:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_
WITH_CHACHA20_POLY1305,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1
305,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
```

El nombre de `AntreaConfig` debe tener el mismo formato que `<cluster-name>-antrea-package`. El nombre de `VSphereCPIConfig` debe tener el formato `<cluster-name>-vsphere-cpi-package`.

Una vez que se hayan creado los archivos de configuración, se crea el objeto de especificación del clúster que hace referencia a los archivos de configuración. Durante la creación del clúster, los archivos de configuración se utilizarán para aprovisionar el clúster y sobrescribir la configuración predeterminada.

## Crear una red de pods enrutables: configuración de Supervisor

Para crear una red de pods enrutables se requiere la configuración en el Supervisor y en el clúster de TKG.

**Nota** El Supervisor debe configurarse con NSX para usar redes de pods enrutables. No se pueden utilizar pods enrutables con redes de VDS.

Para configurar una red de pods enrutables en Supervisor:

1 Cree un nuevo espacio de nombres de vSphere.

   Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

2 Seleccione la opción de casilla de verificación para **Anular configuración de red de supervisor**.

   Consulte Anular la configuración de la red de cargas de trabajo para un espacio de nombres de vSphere para obtener instrucciones.

3 Configure la red de pods enrutables de la siguiente manera.

| Campo | Descripción |
| --- | --- |
| **Modo NAT** | Anule la selección de esta opción para deshabilitar la traducción de direcciones de red (NAT). |
| **Red de espacio de nombres** | Rellene este campo con una subred IP enrutable con el formato Dirección IP/Bits (p. ej., 10.0.0.6/16).<br>NCP creará uno o varios grupos de direcciones IP a partir de los bloques de IP especificados para la red.<br>Como mínimo, debe especificar un tamaño de subred /23. Por ejemplo, si especifica una subred enrutable /23 con un prefijo de subred /28, obtendrá 32 subredes que deberían ser suficientes para un clúster de 6 nodos. Una subred /24 con un prefijo /28 solo obtendrá 2 subredes, que no son suficientes.<br>**Atención** Asegúrese de que la subred de IP enrutable que agregue no se superponga con el **CIDR de servicios** que asigna las direcciones IP para los nodos de clúster. Puede comprobar el **CIDR de servicios** en **Supervisor > Configurar > Red > Red de cargas de trabajo**. |
| **Prefijo de subred de espacio de nombres** | Especifique un prefijo de subred con el formato /28, por ejemplo.<br>El prefijo de subred se utiliza para dividir la subred del pod para cada nodo de la red de espacio de nombres. |

4 Haga clic en **Crear** para crear la red de pods enrutables.

## Crear una red de pods enrutables: configuración de un clúster de TKG

El siguiente ejemplo de YAML muestra cómo configurar un clúster v1beta1 con una red de pods enrutables.

Como se muestra en el siguiente ejemplo, debe eliminar la sección `spec.clusterNetwork.pod` de la especificación del clúster, ya que cloud-provider-vsphere asignará las direcciones IP del pod.

**Nota** El ejemplo se proporciona como un único archivo YAML, pero se puede separar en archivos individuales. Si hace esto, debe crearlos en orden: primero los recursos personalizados de `AntreaConfig` y `VSphereCPIConfig` y después el clúster de target-cluster.

```
---
apiVersion: cni.tanzu.vmware.com/v1alpha1
kind: AntreaConfig
metadata:
 name: target-cluster-antrea-package
spec:
 antrea:
   config:
     defaultMTU: ""
     disableUdpTunnelOffload: false
     featureGates:
       AntreaPolicy: true
       AntreaProxy: true
       AntreaTraceflow: true
       Egress: false
       EndpointSlice: true
       FlowExporter: false
       NetworkPolicyStats: false
       NodePortLocal: false
     noSNAT: true
     tlsCipherSuites:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WI
TH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384
     trafficEncapMode: noEncap
---
apiVersion: cpi.tanzu.vmware.com/v1alpha1
kind: VSphereCPIConfig
metadata:
 name: target-cluster-vsphere-cpi-package
spec:
 vsphereCPI:
   antreaNSXPodRoutingEnabled: true
   insecure: false
   mode: vsphereParavirtualCPI
   tlsCipherSuites:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_
WITH_CHACHA20_POLY1305,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1
305,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
---
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
```

```
 name: target-cluster
spec:
 clusterNetwork:
   services:
     cidrBlocks: ["198.51.100.0/12"]
   serviceDomain: "cluster.local"
 topology:
   class: tanzukubernetescluster
   version: v1.25.7---vmware.3-fips.1-tkg.1
   controlPlane:
     replicas: 3
   workers:
     machineDeployments:
       - class: node-pool
         name: node-pool-1
         replicas: 3
   variables:
     - name: vmClass
       value: guaranteed-medium
     - name: storageClass
       value: tkg2-storage-policy
```

# Ejemplo de v1beta1: clúster con certificados de CA de confianza adicionales para SSL/TLS

Consulte este ejemplo para aprovisionar un clúster v1beta1 con uno o varios certificados de CA de confianza adicionales.

## Ejemplo de v1beta1: clúster con certificados de CA de confianza adicionales

El API de clúster v1beta1 proporciona la variable trust para aprovisionar un clúster con uno o varios certificados de CA de confianza adicionales.

### Tabla 7-1. Variable de confianza de la API v1beta1

| Campo | Descripción |
| --- | --- |
| `trust` | Marcador de sección. No acepta datos. |
| `additionalTrustedCAs` | Marcador de sección. Incluye una matriz de certificados con el `name` para cada uno. |
| `name` | El nombre definido por el usuario para el campo de asignación de `data` en el secreto de Kubernetes que contiene el certificado de CA en formato PEM con codificación base64 doble.<br>**Nota** Se requiere doble codificación base64. Si el contenido no tiene doble codificación base6, no se puede procesar el archivo PEM resultante. |

El siguiente ejemplo demuestra cómo agregar el secreto de Kubernetes que contiene un certificado de CA a una especificación de clúster de la API v1beta1.

```
#cluster-with-trusted-private-reg-cert.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster01
  namespace: tkgs-cluster-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.52.100.0/12"]
    pods:
      cidrBlocks: ["192.101.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.26.5+vmware.2-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-01
          replicas: 3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkgs-storage-profile
      - name: defaultStorageClass
        value: tkgs-storage-profile
      - name: trust
        value:
          additionalTrustedCAs:
          - name: additional-ca-1
```

El siguiente ejemplo muestra el secreto de Kubernetes que incluye un certificado de CA de confianza adicional.

```
#additional-ca-1.yaml
apiVersion: v1
data:
  additional-ca-1: TFMwdExTMUNSGlSzZ3Jaa...VVNVWkpRMEMwdExTMHRDZz09
kind: Secret
metadata:
  name: cluster01-user-trusted-ca-secret
  namespace: tkgs-cluster-ns
type: Opaque
```

Donde:

- El valor de la asignación de `data` del secreto es una cadena definida por el usuario que es el nombre del certificado de CA (`additional-ca-1` en este ejemplo), cuyo valor es el certificado de CA en formato PEM con codificación base64 doble.
- En la sección `metadata`, el secreto debe tener el nombre *`CLUSTER-NAME`*`-user-trusted-ca-secret`, donde *CLUSTER-NAME* es el nombre del clúster. Este secreto se debe crear en el mismo espacio de nombres de vSphere que el clúster.

Para codificar doble en base64 el contenido del certificado de CA.

- Linux: `base64 -w 0 ca.crt | base64 -w 0`
- Windows: https://www.base64encode.org/.

## Procedimiento: Nuevo clúster

Complete el siguiente procedimiento para incluir uno o varios certificados de CA de confianza adicionales en un nuevo clúster de TKGS.

1 Doble codificación base64 del contenido de un certificado de CA.

2 Cree un secreto de Kubernetes que contenga el nombre de la asignación de datos cuyo valor sea un certificado de CA en formato PEM con doble codificación base64.

3 En la especificación del clúster, rellene la variable `trust.additionalTrustedCAs` con el nombre de la asignación de datos.

4 Aprovisione el clúster como lo haría normalmente.

   Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

5 Cuando el clúster se aprovisiona correctamente, el certificado de CA que agregó es de confianza para el clúster.

## Procedimiento: Clúster existente

Complete el siguiente procedimiento para agregar uno o varios certificados de CA de confianza adicionales a un clúster existente.

1 Doble codificación base64 del contenido de un certificado de CA.

2 Cree un secreto de Kubernetes que contenga el nombre de la asignación de datos cuyo valor sea un certificado de CA en formato PEM con doble codificación base64.

3 Compruebe que configuró kubectl editing.

   Consulte Configurar un editor de texto para Kubectl.

4 Edite la especificación del clúster.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-name
```

5 Agregue la sección `trust.additionalTrustedCAs` a la especificación.

6 Rellene el campo de `additionalTrustedCAs` con el nombre de la asignación de datos en el secreto que contiene el certificado de CA en formato PEM con codificación base64 doble

7 Guarde los cambios en el editor de texto y compruebe que kubectl los haya registrado.

```
kubectl edit cluster/tkgs-cluster-name
cluster.run.tanzu.vmware.com/tkgs-cluster-name edited
```

8 Cuando se inicia una actualización gradual para el clúster, se agregan los certificados de CA de confianza adicionales.

   Consulte Información sobre el modelo de actualización gradual para clústeres de Servicio TKG.

### Comprobar los certificados de CA de confianza adicionales

Los certificados de CA de confianza adicionales agregados al clúster se incluyen en el archivo kubeconfig del clúster.

### Rotar el certificado

Para rotar un certificado, cree un nuevo secreto y edite la especificación del clúster con el valor adecuado. Esto activará una actualización gradual del clúster.

**Nota** El sistema no supervisa los cambios en `CLUSTER-NAME-user-trusted-ca-secret`. Si cambia el valor de asignación de `data`, estos cambios no se verán reflejados en el clúster. Debe crear un nuevo secreto y su asignación de datos `name` a `trust.additionalTrustCAs`.

### Solucionar los problemas de certificados de CA de confianza adicionales

Consulte Solucionar errores con certificados de CA de confianza adicionales.

### Caso práctico

El caso práctico más común es para agregar una CA de confianza adicional para conectarse a un registro de contenedor. Consulte Integrar clústeres de Servicio TKG con un registro de contenedor privado.

## Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U2 y versiones posteriores)

Consulte estas instrucciones para aprovisionar un clúster de TKG basado en una ClusterClass personalizada. Tenga en cuenta que estas instrucciones son específicas de los entornos de vSphere 8 U2 y versiones posteriores.

### Requisitos previos

El procedimiento para aprovisionar un clúster de TKG basado en una ClusterClass personalizada se actualiza para la versión vSphere 8 U2.

Debe cumplir los siguientes requisitos previos.

- Entorno de vSphere 8 U2 y versiones posteriores
- Administración de cargas de trabajo habilitada
- Supervisor configurados
- Cliente de Ubuntu con Herramientas de la CLI de Kubernetes para vSphere instalado

**Atención** La ClusterClass personalizada es una función experimental de Kubernetes según la documentación de la API del clúster ascendente. Debido a la variedad de personalizaciones disponibles con la ClusterClass personalizada, VMware no puede probar ni validar todas las personalizaciones posibles. Los clientes son responsables de probar, validar y solucionar los problemas que puedan surgir en sus clústeres ClusterClass personalizados. Los clientes pueden abrir tickets de soporte relacionados con sus clústeres ClusterClass personalizados. Sin embargo, el soporte de VMware tan solo puede ayudar dentro de unos límites y no puede garantizar la resolución de todos los problemas que se abran para los clústeres ClusterClass personalizados. Los clientes deben tener en cuenta estos riesgos antes de implementar clústeres ClusterClass personalizados en entornos de producción.

## Flujos de trabajo de alto nivel

Los flujos de trabajo de alto nivel son los siguientes.

El siguiente flujo de trabajo es todo lo que necesita para comenzar.

| Paso | Tarea | Instrucciones |
| --- | --- | --- |
| 1 | Cree una ClusterClass personalizada mediante la clonación de la ClusterClass predeterminada. | 1: Crear una ClusterClass personalizada |
| 2 | Aprovisione un nuevo clúster de TKG según la ClusterClass personalizada y compruebe que todos los nodos del clúster aparezcan correctamente. | 2: Crear un clúster de TKG basado en la ClusterClass personalizada |

Consulte el siguiente flujo de trabajo para realizar cambios en la ClusterClass personalizada e inicie una actualización gradual de los nodos del clúster ClusterClass personalizado.

**Nota** La operación que se muestra en el siguiente flujo de trabajo es un ejemplo de lo que se puede hacer con una ClusterClass personalizada. Los casos prácticos pueden ser diferentes, pero se debería poder aplicar el flujo de trabajo general.

| Paso | Tarea | Instrucciones |
| --- | --- | --- |
| 3 | Acceda mediante SSH a uno de los nodos de trabajo para confirmar que hay paquetes que se deben actualizar. | 3: Comprobar la existencia de actualizaciones de paquetes |
| 4 | Actualice la ClusterClass personalizada con un nuevo comando que realice las actualizaciones. | 4: Actualizar la ClusterClass personalizada |
| 5 | Confirme la implementación de nodos nuevos con las actualizaciones ya ejecutadas. | 5: Comprobar la actualización gradual de los nodos del clúster |

## 1: Crear una ClusterClass personalizada

La primera parte implica la creación de una ClusterClass personalizada denominada `ccc` (abreviatura de `customclusterclass`) mediante la clonación de la ClusterClass predeterminada, que se denomina `tanzukubernetescluster`.

**Nota** El nombre de ClusterClass personalizado lo define el usuario. Si utiliza un nombre diferente, ajuste las instrucciones según corresponda.

1 Cree y configure un espacio de nombres de vSphere denominado *ccc-ns*.

   Configure los permisos, el almacenamiento, la biblioteca de contenido y las clases de máquinas virtuales. Consulte la Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG según sea necesario.

   **Nota** El nombre de espacio de nombres de vSphere lo define el usuario. Si utiliza un nombre diferente, ajuste las instrucciones según corresponda.

2 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USER@vsphere.local
```

3 Escriba el resultado de la ClusterClass predeterminada en un archivo denominado *ccc.yaml*.

```
kubectl -n ccc-ns get clusterclass tanzukubernetescluster -o yaml > ccc.yaml
```

   O bien, la versión del acceso directo:

```
kubectl -n ccc-ns get cc tanzukubernetescluster -o yaml > ccc.yaml
```

4 Abra el archivo de la ClusterClass clonada para editarlo.

```
vim ccc.yaml
```

5 Edite el archivo `ccc.yaml`.

   - Elimine la línea `metadata.creationTimestamp`
   - Elimine la línea `metadata.generation`
   - Elimine la línea `metadata.resourceVersion`
   - Elimine la línea `metadata.uid`
   - Cambie el valor de `metadata.name` en `tanzukubernetescluster` a `ccc`
   - Deje el valor de `metadata.namespace` tal como está: `ccc-ns`
   - Deje el valor de `metadata.annotations` tal como está para `run.tanzu.vmware.com/resolve-tkr: ""`. Esta anotación es necesaria para los datos o la resolución de TKR.

6 Guarde y compruebe los cambios.

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: ClusterClass
metadata:
  annotations:
    run.tanzu.vmware.com/resolve-tkr: ""
  name: ccc
  namespace: ccc-ns
spec:
...
```

7 Cree el objeto de ClusterClass personalizado.

```
kubectl apply -f ccc.yaml -n ccc-ns
```

Resultado esperado:

```
clusterclass.cluster.x-k8s.io/ccc created
```

8 Enumere la ClusterClass personalizada.

```
kubectl get cc -n ccc-ns
```

Resultado esperado:

```
NAME                        AGE
ccc                         3m14s
tanzukubernetescluster      29m
```

## 2: Crear un clúster de TKG basado en la ClusterClass personalizada

Utilice la API v1beta1 del clúster para crear un clúster basado en una ClusterClass.

1 Cree el manifiesto de `ccc-cluster.yaml` para aprovisionar el clúster.

```
#ccc-cluster.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: ccc-cluster
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 192.0.2.0/16
    services:
      cidrBlocks:
      - 198.51.100.0/12
    serviceDomain: cluster.local
  topology:
    class: ccc
    version: v1.26.5---vmware.2-fips.1-tkg.1
    controlPlane:
```

```
      replicas: 1
    workers:
      machineDeployments:
        - class: node-pool
          name: tkgs-node-pool-1
          replicas: 1
    variables:
    - name: vmClass
      value: guaranteed-small
    - name: storageClass
      value: tkg-storage-profile
```

Donde:

- El valor de `metadata.name` es el nombre del clúster: `ccc-cluster`
- El valor de `spec.topology.class` es el nombre de la ClusterClass personalizada: `ccc`
- El valor de `spec.topology.version` es la versión de TKR
- El valor de `spec.topology.variables.storageClass` es el nombre de la clase de almacenamiento persistente

**Nota** Para fines de prueba, 1 réplica es suficiente para el plano de control y el grupo de nodos de trabajo. En producción, utilice 3 réplicas para el plano de control y al menos 3 réplicas para cada grupo de nodos de trabajo.

2 Cree el clúster de TKG en función de la ClusterClass personalizada.

```
kubectl apply -f ccc-cluster.yaml -n ccc-ns
```

Resultado esperado:

```
cluster.cluster.x-k8s.io/ccc-cluster created
```

3 Compruebe el aprovisionamiento del clúster.

Ejecute el siguiente comando. Espere a que todos los nodos del clúster aparezcan correctamente.

```
kubectl -n ccc-ns get
cc,clusters,vsphereclusters,kcp,machinedeployment,machineset,machine,vspheremachine,virtual
machineservice
```

**Nota** Resultará útil ejecutar este comando en una sesión independiente para que pueda supervisar el progreso de la actualización gradual en el paso 5.

## 3: Comprobar la existencia de actualizaciones de paquetes

Acceda mediante SSH a uno de los nodos de trabajo para confirmar que hay paquetes que se deben actualizar.

**Nota** El objetivo de este paso simplemente es confirmar que hay paquetes para actualizar, no actualizarlos realmente. La ClusterClass personalizada los actualizará cuando se implementen nodos del clúster nuevos (los pasos que siguen a continuación). Este paso y los que se indican a continuación deben ser un ejemplo de lo que se puede hacer con una ClusterClass personalizada.

1 Ejecute el siguiente comando para obtener el secreto de SSH.

```
export CC=ccc-cluster && kubectl get secret -n ccc-ns ${CC}-ssh -o jsonpath={.data.ssh-privatekey} | base64 -d > ${CC}-ssh && chomd 4000 ${CC}-ssh
```

2 Ejecute el siguiente comando para obtener la dirección IP de la máquina virtual del nodo de trabajo.

```
kubectl -n ccc-ns get vm -o wide
```

**Nota** Si implementó varios nodos de trabajo, elija uno. No utilice un nodo de plano de control.

3 Ejecute el siguiente comando para utilizar SSH en la máquina virtual del nodo de trabajo.

```
ssh -i ${CC}-ssh vmware-system-user@IP-ADDRESS-OF-WORKER-NODE
```

Por ejemplo:

```
ssh -i ${CC}-ssh vmware-system-user@192.168.128.55
```

**Nota** Introduzca "yes" para continuar con la conexión.

Resultado esperado: después de utilizar SSH en el host, debería ver el siguiente mensaje.

```
tdnf update info not availble yet!
```

4 Ejecute los siguientes comandos y compruebe si hay actualizaciones.

```
sudo -i
```

```
tdnf update
```

5 En el símbolo del sistema, introduzca "N" para no (no actualizar).

Resultado esperado:

```
Operation aborted
```

**Nota** El propósito aquí es simplemente comprobar si hay actualizaciones, no iniciar las actualizaciones. Para iniciar las actualizaciones, agregue un comando a la ClusterClass personalizada en la siguiente sección.

6 Escriba "exit" para cerrar la sesión de SSH y, a continuación, vuelva a escribir "exit".

## 4: Actualizar la ClusterClass personalizada

Actualice la ClusterClass personalizada con un comando nuevo que realice una actualización de tdnf.

1 Abra la ClusterClass personalizada denominada `ccc` para editarla.

```
kubectl edit cc ccc -n ccc-ns
```

2 Desplácese hacia abajo hasta la siguiente sección con `postKubeadmCommands`.

```
  - definitions:
    - jsonPatches:
      - op: add
        path: /spec/template/spec/kubeadmConfigSpec/postKubeadmCommands
        valueFrom:
          template: |
            - touch /root/kubeadm-complete
            - vmware-rpctool 'info-set guestinfo.kubeadm.phase complete'
            - vmware-rpctool 'info-set guestinfo.kubeadm.error ---'
      selector:
        apiVersion: controlplane.cluster.x-k8s.io/v1beta1
        kind: KubeadmControlPlaneTemplate
        matchResources:
          controlPlane: true
    - jsonPatches:
      - op: add
        path: /spec/template/spec/postKubeadmCommands
        valueFrom:
          template: |
            - touch /root/kubeadm-complete
            - vmware-rpctool 'info-set guestinfo.kubeadm.phase complete'
            - vmware-rpctool 'info-set guestinfo.kubeadm.error ---'
      selector:
        apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
        kind: KubeadmConfigTemplate
        matchResources:
          machineDeploymentClass:
            names:
            - node-pool
    name: controlPlanePostKubeadmCommandsSuccess
```

Agregue el siguiente comando en ambos campos `valueFrom.template`.

```
- tdnf update -y
```

Por ejemplo:

```
  - definitions:
    - jsonPatches:
      - op: add
        path: /spec/template/spec/kubeadmConfigSpec/postKubeadmCommands
        valueFrom:
          template: |
            - touch /root/kubeadm-complete
            - vmware-rpctool 'info-set guestinfo.kubeadm.phase complete'
            - vmware-rpctool 'info-set guestinfo.kubeadm.error ---'
            - tdnf update -y
      selector:
        apiVersion: controlplane.cluster.x-k8s.io/v1beta1
        kind: KubeadmControlPlaneTemplate
        matchResources:
          controlPlane: true
    - jsonPatches:
      - op: add
        path: /spec/template/spec/postKubeadmCommands
        valueFrom:
          template: |
            - touch /root/kubeadm-complete
            - vmware-rpctool 'info-set guestinfo.kubeadm.phase complete'
            - vmware-rpctool 'info-set guestinfo.kubeadm.error ---'
            - tdnf update -y
      selector:
        apiVersion: bootstrap.cluster.x-k8s.io/v1beta1
        kind: KubeadmConfigTemplate
        matchResources:
          machineDeploymentClass:
            names:
            - node-pool
    name: controlPlanePostKubeadmCommandsSuccess
```

3 Guarde los cambios en la ClusterClass personalizada y cierre el editor.

```
wq
```

Resultado esperado:

```
clusterclass.cluster.x-k8s/ccc edited
```

## 5: Comprobar la actualización gradual de los nodos del clúster

La actualización de ClusterClass personalizada activa una actualización gradual de los nodos del clúster para los clústeres aprovisionados en función de esa ClusterClass. Los nodos nuevos aparecen cuando se aplica el comando anterior.

1 Para comprobar el aprovisionamiento del clúster, ejecute el siguiente comando.

   Espere a que todos los nodos del clúster aparezcan correctamente.

   ```
   kubectl -n ccc-ns get
   cc,clusters,vsphereclusters,kcp,machinedeployment,machineset,machine,vspheremachine,virtual
   machineservice
   ```

2 Debería ver que se implementen nodos nuevos con UUID nuevos.

3 Ejecute el siguiente comando para utilizar SSH en la máquina virtual del nodo de trabajo.

   ```
   ssh -i ${CC}-ssh vmware-system-user@IP-ADDRESS-OF-WORKER-NODE
   ```

   Resultado esperado: después de utilizar SSH en el host, debería ver el siguiente mensaje.

   ```
   tdnf update info not availble yet!
   ```

4 Ejecute los siguientes comandos.

   ```
   sudo -i
   ```

   ```
   tdnf update
   ```

   Resultado esperado: debería ver muchos menos paquetes que deben actualizarse.

5 En el símbolo del sistema, introduzca "N" para no (no actualizar).

   Resultado esperado:

   ```
   Operation aborted
   ```

6 Ejecute el siguiente comando para confirmar que tdnf se ejecutó.

   ```
   cat /var/log/cloud-init-output.log | grep -i tdnf
   ```

7 Escriba "exit" para cerrar la sesión de SSH y, a continuación, vuelva a escribir "exit".

## Mantenimiento de una ClusterClass personalizada

Después de actualizar el Servicio TKG a una nueva versión, debe asegurarse de que la ClusterClass personalizada derivada de la ClusterClass predeterminada de la versión de Servicio TKG anterior se actualice con los cambios para la ClusterClass predeterminada que se enviaron con la nueva versión de Servicio TKG.

Utilice el siguiente flujo de trabajo para mantener la ClusterClass personalizada sincronizada con la ClusterClass proporcionada por el sistema. Tenga en cuenta que estas instrucciones asumen que creó una ClusterClass personalizada inicial como se describe aquí.

1 Actualizar la versión de Servicio TKG.

   Por ejemplo, actualice de Servicio TKG v3.0 a v3.1.

   Consulte Capítulo 3 Instalar y actualizar Servicio TKG.

2 Cree una nueva ClusterClass personalizada siguiendo las Flujos de trabajo de alto nivel de este documento.

   Cree manualmente una versión de la nueva ClusterClass personalizada anexando a su nombre la versión de Servicio TKG, como `ccc-3.1`.

3 Agregue las variables y las revisiones personalizadas de la ClusterClass personalizada anterior a la nueva ClusterClass personalizada.

   Para ello, abra `cat ccc.yaml` y copie las variables y las revisiones personalizadas desde ese archivo a `ccc-3.1.yaml`.

4 Aplique la nueva ClusterClass personalizada y espere a que la reconciliación se realice correctamente.

5 Para actualizar los clústeres de TKG mediante la ClusterClass personalizada anterior al nuevo ClusterClass personalizado, edite el campo `spec.topology.class` en el objeto Cluster.

## ClusterClass no administrada

Para vSphere 8 U2 y posterior, hay una anotación que puede agregar a una ClusterClass personalizada si no desea que la controladora TKG la administre. Tenga en cuenta que, si agrega esta anotación, será responsable de crear manualmente todos los objetos de Kubernetes subyacentes, como certificados, secretos, etc. Consulte la Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U1) de ClusterClass personalizada de vSphere 8 U1 para obtener instrucciones sobre cómo hacerlo.

La anotación es la siguiente:

| Clave de anotación | Valor |
| --- | --- |
| run.tanzu.vmware.com/unmanaged-clusterclass | " " |

A continuación, se muestra un ejemplo de cómo agregaría la anotación a la ClusterClass personalizada denominada `ccc`:

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: ClusterClass
metadata:
  annotations:
    run.tanzu.vmware.com/resolve-tkr: ""
    run.tanzu.vmware.com/unmanaged-clusterclass: ""
```

```
  name: ccc
  namespace: ccc-ns
spec:
...
```

## Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U1)

Consulte estas instrucciones para aprovisionar un clúster de TKG basado en una ClusterClass personalizada. Tenga en cuenta que estas instrucciones son específicas para entornos de vSphere 8 U1.

### Requisitos previos

El procedimiento para aprovisionar un clúster de TKG basado en una ClusterClass personalizada está disponible a partir de la versión vSphere 8 U1. Si utiliza vSphere 8 U2, consulte Ejemplo de v1beta1: clúster basado en una ClusterClass personalizada (flujo de trabajo de vSphere 8 U2 y versiones posteriores).

Debe cumplir los siguientes requisitos previos.

- Entorno de vSphere 8 U1
- Administración de cargas de trabajo habilitada
- Supervisor configurados
- Cliente de Ubuntu con Herramientas de la CLI de Kubernetes para vSphere instalado

---

**Atención** La ClusterClass personalizada es una función experimental de Kubernetes según la documentación de la API del clúster ascendente. Debido a la variedad de personalizaciones disponibles con la ClusterClass personalizada, VMware no puede probar ni validar todas las personalizaciones posibles. Los clientes son responsables de probar, validar y solucionar los problemas que puedan surgir en sus clústeres ClusterClass personalizados. Los clientes pueden abrir tickets de soporte relacionados con sus clústeres ClusterClass personalizados. Sin embargo, el soporte de VMware tan solo puede ayudar dentro de unos límites y no puede garantizar la resolución de todos los problemas que se abran para los clústeres ClusterClass personalizados. Los clientes deben tener en cuenta estos riesgos antes de implementar clústeres ClusterClass personalizados en entornos de producción.

---

### Parte 1: Crear el ClusterClass personalizado

La primera parte implica la creación de una ClusterClass personalizada mediante la clonación de la ClusterClass predeterminada, que se denomina `tanzukubernetescluster`.

1. Cree un espacio de nombres de vSphere denominado *custom-ns*.
2. Inicie sesión en Supervisor.
3. Cambie el contexto al espacio de nombres de vSphere denominado *custom-ns*.

4 Obtenga la ClusterClass predeterminada.

```
kubectl get clusterclass tanzukubernetescluster -o json
```

5 Cree una ClusterClass personalizada denominada *custom-cc* mediante la clonación de la ClusterClass predeterminada.

```
kubectl get clusterclass tanzukubernetescluster -o json | jq '.metadata.name="custom-cc"'
| kubectl apply -f -
```

Resultado esperado:

```
clusterclass.cluster.x-k8s.io/custom-cc created
```

6 Obtenga la ClusterClass personalizada.

```
kubectl get clusterclass custom-cc -o json
```

Si fuera necesario, puede utilizar "less" para ver la ClusterClass personalizada.

```
kubectl get clusterclass custom-cc -o json | less
```

**Nota** Introduzca el comando "q" para salir de "less".

## Parte 2: Crear objetos de Supervisor necesarios para aprovisionar el clúster de TKG

La siguiente parte consiste en crear los objetos de Supervisor necesarios para la implementación inicial de un clúster de TKG personalizado mediante la ClusterClass personalizada.

**Nota** De forma predeterminada, se utiliza el nombre de clúster "ccc-cluster". Si utiliza un nombre de clúster diferente, deberá cambiarlo en los campos correspondientes.

1 Cree el emisor del certificado de extensiones autofirmado.

```
#self-signed-extensions-issuer.yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: self-signed-extensions-issuer
spec:
  selfSigned: {}
```

```
kubectl apply -f self-signed-extensions-issuer.yaml -n custom-ns
```

Resultado esperado:

```
issuer.cert-manager.io/self-signed-extensions-issuer created
```

2 Cree el secreto para el certificado de CA de las extensiones.

```
#extensions-ca-certificate.yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ccc-cluster-extensions-ca
spec:
  commonName: kubernetes-extensions
  duration: 87600h0m0s
  isCA: true
  issuerRef:
    kind: Issuer
    name: self-signed-extensions-issuer
  secretName: ccc-cluster-extensions-ca
  usages:
  - digital signature
  - cert sign
  - crl sign
```

```
kubectl apply -f extensions-ca-certificate.yaml -n custom-ns
```

Resultado esperado:

```
certificate.cert-manager.io/ccc-cluster-extensions-ca created
```

3 Cree el emisor del certificado de CA de las extensiones.

```
#extensions-ca-issuer.yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: ccc-cluster-extensions-ca-issuer
spec:
  ca:
    secretName: ccc-cluster-extensions-ca
```

```
kubectl apply -f extensions-ca-issuer.yaml -n custom-ns
```

Resultado esperado:

```
issuer.cert-manager.io/ccc-cluster-extensions-ca-issuer created
```

4 Cree el secreto para el certificado del servicio de autenticación.

```
#auth-svc-cert.yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ccc-cluster-auth-svc-cert
spec:
  commonName: authsvc
  dnsNames:
```

```
  - authsvc
  - localhost
  - 127.0.0.1
  duration: 87600h0m0s
  issuerRef:
    kind: Issuer
    name: ccc-cluster-extensions-ca-issuer
  secretName: ccc-cluster-auth-svc-cert
  usages:
  - server auth
  - digital signature
```

```
kubectl apply -f auth-svc-cert.yaml -n custom-ns
```

Resultado esperado:

```
certificate.cert-manager.io/ccc-cluster-auth-svc-cert created
```

5 Compruebe la creación de los emisores y los certificados.

```
kubectl get issuers -n custom-ns
NAME                              READY   AGE
ccc-cluster-extensions-ca-issuer  True    2m57s
self-signed-extensions-issuer     True    14m
```

```
kubectl get certs -n custom-ns
NAME                         READY   SECRET                       AGE
ccc-cluster-auth-svc-cert    True    ccc-cluster-auth-svc-cert    34s
ccc-cluster-extensions-ca    True    ccc-cluster-extensions-ca    5m
```

## Parte 3: Crear un clúster de TKG basado en la ClusterClass personalizada

Utilice la API v1beta1 del clúster para crear un clúster basado en una ClusterClass. Un clúster v1beta1 basado en una ClusterClass personalizada requiere el siguiente conjunto mínimo de variables.

| Variable | Descripción |
| --- | --- |
| vmClass | Consulte Uso de clases de máquinas virtuales con clústeres de Servicio TKG. |
| storageClass | Consulte Configurar el almacenamiento persistente para el espacio de nombres de vSphere. |
| ntp | Servidor NTP utilizado para habilitar Supervisor. |

| Variable | Descripción |
| --- | --- |
| extensionCert | Se genera automáticamente después de crear el "certificado de CA de extensiones" en la sección anterior. |
| clusterEncryptionConfigYaml | la siguiente sección revisará el proceso de obtención de este archivo |

1 Cree el secreto de cifrado.

```
#encryption-secret.yaml
apiVersion: v1
data:
  key: all3dzZpODFmRmh6MVlJbUtQQktuN2ViQzREbDBQRHlxVk8yYXRxTW9QQT0=
kind: Secret
metadata:
  name: ccc-cluster-encryption
type: Opaque
```

```
kubectl apply -f encryption-secret.yaml -n custom-ns
```

Resultado esperado:

```
secret/ccc-cluster-encryption created
```

2 Recopile el servidor NTP de Supervisor.

```
kubectl -n vmware-system-vmop get configmap vmoperator-network-config -o
jsonpath={.data.ntpservers}
```

3 Cree el manifiesto de `cluster-with-ccc.yaml` para aprovisionar el clúster.

```
#cluster-with-ccc.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: ccc-cluster
spec:
  clusterNetwork:
    pods:
      cidrBlocks:
      - 193.0.0.0/16
    serviceDomain: managedcluster1.local
    services:
      cidrBlocks:
      - 198.201.0.0/16
  topology:
    class: custom-cc
    version: v1.26.5---vmware.2-fips.1-tkg.1
    controlPlane:
      metadata: {}
      replicas: 3
    workers:
      machineDeployments:
```

```
      - class: node-pool
        metadata: { }
        name: node-pool-workers
        replicas: 3
    variables:
    - name: vmClass
      value: guaranteed-medium
    - name: storageClass
      value: tkg-storage-profile
    - name: ntp
      value: time.acme.com
    - name: extensionCert
      value:
        contentSecret:
          key: tls.crt
          name: ccc-cluster-extensions-ca
    - name: clusterEncryptionConfigYaml
      value: LS0tCm...Ht9Cg==
```

En el manifiesto del clúster, compruebe o actualice los siguientes campos:

| Parámetro | Descripción |
| --- | --- |
| `metadata.name` | Nombre del clúster v1beta1. |
| `spec.topology.class` | Nombre de la ClusterClass personalizada. |
| `spec.topology.version` | Versión de versión de Tanzu Kubernetes |
| `spec.topology.variables.storageClass.value` | StoragePolicy asociada al espacio de nombres de vSphere en el que se aprovisionará el clúster |
| `spec.topology.variables.ntp.value` | Dirección del servidor NTP |
| `spec.topology.variables.extensionCert.value.contentSecret.name` | Verificar |
| `spec.topology.variables.clusterEncryptionConfigYaml.value` | Rellene con el valor de `data.key` del secreto ClusterEncryptionConfig |

4 Cree el clúster en función de la ClusterClass personalizada.

```
kubectl apply -f cluster-with-ccc.yaml -n custom-ns
```

Resultado esperado:

```
cluster.cluster.x-k8s.io/ccc-cluster created
```

Mediante vSphere Client, compruebe que se ha creado el clúster.

5 Inicie sesión en el clúster de TKG.

```
kubectl vsphere login --server=xxx.xxx.xxx.xxx --vsphere-username USERNAME@vsphere.local
--tanzu-kubernetes-cluster-name ccc-cluster --tanzu-kubernetes-cluster-namespace custom-ns
```

## Parte 4: Crear objetos de Supervisor necesarios para administrar el clúster de TKG

Una vez que se aplica el clúster con CCC, varios controladores intentarán aprovisionarlo; no obstante, los recursos de infraestructura subyacentes aún requieren objetos adicionales para arrancar correctamente.

| Parámetro | Valor |
| --- | --- |
| Autenticación | Los valores de autenticación deben recopilarse y actualizarse en un archivo denominado values.yaml |
| Codificado en Base64 | El archivo values.yaml se codificará en forma de cadena en base64 |
| guest-cluster-auth-service-data-values.yaml | Esta cadena se agregará al archivo guest-cluster-auth-service-data-values.yaml descargado de CCC_config_yamls.tar.gz antes de aplicar el archivo |
| Secreto GuestClusterAuthSvcDataValues | Por último, el arranque del clúster invitado debe modificarse para que haga referencia al secreto GuestClusterAuthSvcDataValues recién creado. |

1 Cambie el contexto a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster.

```
kubectl config use-context custom-ns
```

2 Obtenga el valor de `authServicePublicKeys`.

```
kubectl -n vmware-system-capw get configmap vc-public-keys -o
jsonpath="{.data.vsphere\.local\.json}"
```

Copie el resultado en un archivo de texto denominado `values.yaml`.

```
authServicePublicKeys: '{"issuer_url":"https://...SShrDw=="]}]}}'
```

3 Obtenga el UID del clúster para actualizar las `authServicePublicKeys`.

```
kubectl get cluster -n custom-ns ccc-cluster -o yaml | grep uid
```

4 En la sección `authServicePublicKeys` del archivo `values.yaml`, anexe el UID del clúster al valor "`client_id`".

Sintaxis: `vmware-tes:vc:vns:k8s:`*`clusterUID`*

Por ejemplo:

```
vmware-tes:vc:vns:k8s:7d95b50b-4fd4-4642-82a3-5dbfe87f499c
```

5 Obtenga el valor del certificado (reemplace *ccc-cluster* por el nombre de clúster seleccionado).

```
kubectl -n custom-ns get secret ccc-cluster-auth-svc-cert -o jsonpath="{.data.tls\.crt}" |
base64 -d
```

6 Agregue el certificado a `values.yaml`.

Agregue el contenido del certificado debajo de la sección `authServicePublicKeys`.

**Nota** El certificado debe tener una sangría de 4 espacios para evitar errores.

Por ejemplo:

```
authServicePublicKeys: '{"issuer_url":"https://...SShrDw=="]}]}}'
ceritificate: |
    -----BEGIN CERTIFICATE-----
    MIIDPTCCAiWgAwIBAgIQMibGSjeuJelQoPxCof/+xzANBgkqhkiG9w0BAQsFADAg
    ...
    sESk/RDTB1UAvi8PD3zcbEKZuRxuo4IAJqFFbAabwULhjUo0UwT+dIJo1gLf5/ep
    VoIRJS7j6VT98WbKyZp5B4I=
    -----END CERTIFICATE-----
```

7 Obtenga el valor de privateKey.

```
kubectl -n custom-ns get secret ccc-cluster-auth-svc-cert -o jsonpath="{.data.tls\.key}"
```

8 Compruebe su archivo `values.yaml`.

```
authServicePublicKeys: '{"issuer_url":"https://10.197.79.141/openidconnect/
vsphere.local","client_id":"vmware-tes:vc:vns:k8s:7d95...499c",...SShrDw=="]}]}}'
certificate: |
    -----BEGIN CERTIFICATE-----
    MIIDPTCCAiWgAwIBAgIQWQyXAQDRMhgrGre8ysVN0DANBgkqhkiG9w0BAQsFADAg
    ...
    uJSBP49sF0nKz5nf7w+BdYE=
    -----END CERTIFICATE-----
privateKey: LS0tLS1CRUdJTi...VktLS0tLQo=
```

9 Obtenga el valor hash del archivo `values.yaml` para la codificación en Base64 para recopilar la salida del archivo `guest-cluster-auth-service-data-values.yaml`.

```
base64 -i values.yaml -w 0
```

10 Cree el archivo `guest-cluster-auth-service-data-values.yaml`.

A continuación se muestra una plantilla para el secreto.

```
apiVersion: v1
data:
  values.yaml: YXV0a...ExRbz0K
kind: Secret
metadata:
```

```
  labels:
    tkg.tanzu.vmware.com/cluster-name: ccc-cluster
    tkg.tanzu.vmware.com/package-name: guest-cluster-auth-
service.tanzu.vmware.com.1.3.0+tkg.2-vmware
  name: ccc-cluster-guest-cluster-auth-service-data-values
type: Opaque
```

Consulte la siguiente tabla para rellenar los valores secretos esperados.

| Parámetro | Valor |
| --- | --- |
| `data.values.yaml` | Cadena de `values.yaml` codificada en Base64 |
| `metadata.labels.cluster-name` | Nombre del clúster, como `ccc-cluster` |
| `metadata.labels.package-name` | guest-cluster-auth-service.tanzu.vmware.com.*version*<br>Para obtener este valor, ejecute el comando `kubectl get tkr v1.26.5---vmware.2-fips.1-tkg.1 -o yaml`<br>Cambie la versión de TKR según la versión que esté utilizando |
| `metadata.name` | Nombre del clúster, como `ccc-cluster` |

11 Cree el secreto `guest-cluster-auth-service-data-values.yaml`.

```
kubectl apply -f guest-cluster-auth-service-data-values.yaml -n custom-ns
```

12 Edite el arranque del clúster para hacer referencia al secreto.

```
kubectl edit clusterbootstrap ccc-cluster -n custom-ns
```

13 Agregue las siguientes líneas debajo de la línea `guest-cluster-auth-service.tanzu.vmware.com.`*`version`*`:`.

```

valuesFrom:
  secretRef: ccc-cluster-guest-cluster-auth-service-data-values
```

Por ejemplo:

```
spec:
  additionalPackages:
  - refName: guest-cluster-auth-service.tanzu.vmware.com.1.3.0+tkg.2-vmware
    valuesFrom:
      secretRef: ccc-cluster-guest-cluster-auth-service-data-values
```

14 Guarde y salga para aplicar las modificaciones de clusterbootstrap.

## Parte 5: Configurar la seguridad de pods

Si utiliza TKR 1.25 y versiones posteriores, configure la seguridad de pods para el espacio de nombres de vSphere denominado `custom-ns`. Consulte Configurar PSA para TKR 1.25 y versiones posteriores.

Si utiliza TKR 1.24 y versiones anteriores, los pods del clúster requieren el enlace a la directiva de seguridad de pods. Para aplicar los objetos de recursos necesarios en el nivel de clúster, utilice el siguiente proceso.

1 Recopile el kubeconfig del clúster de TKG.

```
kubectl -n custom-ns get secret ccc-cluster-kubeconfig -o jsonpath="{.data.value}" |
base64 -d > ccc-cluster-kubeconfig
```

2 Cree el archivo `psp.yaml`.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: tanzu-system-kapp-ctrl-restricted
spec:
  privileged: false
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
    - ALL
  volumes:
    - configMap
    - emptyDir
    - projected
    - secret
    - downwardAPI
    - persistentVolumeClaim
  hostNetwork: false
  hostIPC: false
  hostPID: false
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    rule: MustRunAs
    ranges:
      - min: 1
        max: 65535
  fsGroup:
    rule: MustRunAs
    ranges:
      - min: 1
        max: 65535
  readOnlyRootFilesystem: false
```

3 Aplique la directiva de seguridad de pods.

```
KUBECONFIG=ccc-cluster-kubeconfig kubectl apply -f psp.yaml
```

4 Inicie sesión en el clúster de TKG.

```
kubectl vsphere login --server=10.197.154.66 --vsphere-username
administrator@vsphere.local --insecure-skip-tls-verify --tanzu-kubernetes-cluster-name ccc-
cluster --tanzu-kubernetes-cluster-namespace custom-ns
```

5 Enumere los espacios de nombres.

```
KUBECONFIG=ccc-cluster-kubeconfig kubectl get ns -A
```

```
NAME                           STATUS   AGE
default                        Active   13d
kube-node-lease                Active   13d
kube-public                    Active   13d
kube-system                    Active   13d
secretgen-controller           Active   13d
tkg-system                     Active   13d
vmware-system-antrea           Active   13d
vmware-system-cloud-provider   Active   13d
vmware-system-csi              Active   13d
vmware-system-tkg              Active   13d
```

## Parte 6: Sincronizar funciones de SSO de vSphere con el clúster de TKG personalizado

El objeto Rolebinding para los usuarios de vCenter Single Sign-On que están integrados en los espacios de nombres de vSphere debe sincronizarse desde Supervisor con el clúster de TKG para que los desarrolladores administren las cargas de trabajo del clúster.

Este proceso requiere que se exporte la lista de RoleBinding existente desde Supervisor, que se recopilen los RoleBinding que tengan la función "editar" y que se cree el archivo `sync-cluster-edit-rolebinding.yaml` y, a continuación, se aplique al clúster de TKG mediante su KUBECONFIG.

1 Recopile los RoleBindings existentes en Supervisor.

```
kubectl get rolebinding -n custom-ns  -o yaml
```

2 En la lista de objetos RoleBinding devuelta, identifique los que tengan `roleRef.name` igual a "editar".

Por ejemplo:

```
apiVersion: v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: "2023-08-25T18:44:45Z"
    name: ccc-cluster-8lr5x-ccm
    namespace: custom-ns
    ownerReferences:
```

```
    - apiVersion: vmware.infrastructure.cluster.x-k8s.io/v1beta1
      blockOwnerDeletion: true
      controller: true
      kind: ProviderServiceAccount
      name: ccc-cluster-8lr5x-ccm
      uid: b5fb9f01-9a55-4f69-8673-fadc49012994
    resourceVersion: "108766602"
    uid: eb93efd4-ae56-4d9f-a745-d2782885e7fb
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: ccc-cluster-8lr5x-ccm
  subjects:
  - kind: ServiceAccount
    name: ccc-cluster-8lr5x-ccm
    namespace: custom-ns
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: "2023-08-25T18:44:45Z"
    name: ccc-cluster-8lr5x-pvcsi
    namespace: custom-ns
    ownerReferences:
    - apiVersion: vmware.infrastructure.cluster.x-k8s.io/v1beta1
      blockOwnerDeletion: true
      controller: true
      kind: ProviderServiceAccount
      name: ccc-cluster-8lr5x-pvcsi
      uid: d9342f8f-13d2-496d-93cb-b24edfacb5c1
    resourceVersion: "108766608"
    uid: fd1820c7-7993-4299-abb7-bb67fb17f1fd
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: Role
    name: ccc-cluster-8lr5x-pvcsi
  subjects:
  - kind: ServiceAccount
    name: ccc-cluster-8lr5x-pvcsi
    namespace: custom-ns
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: "2023-08-25T16:58:06Z"
    labels:
      managedBy: vSphere
    name: wcp:custom-ns:group:vsphere.local:administrators
    namespace: custom-ns
    resourceVersion: "108714148"
    uid: d74a98c7-e7da-4d71-b1d5-deb60492d429
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: edit
  subjects:
  - apiGroup: rbac.authorization.k8s.io
```

```
    kind: Group
    name: sso:Administrators@vsphere.local
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: "2023-08-25T16:58:21Z"
    labels:
      managedBy: vSphere
    name: wcp:custom-ns:user:vsphere.local:administrator
    namespace: custom-ns
    resourceVersion: "108714283"
    uid: 07f7dbba-2670-4100-a59b-c09e4b2edd6b
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: edit
  subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: sso:Administrator@vsphere.local
kind: List
metadata:
  resourceVersion: ""
```

3 Cree un archivo denominado `sync-cluster-edit-rolebinding.yaml` para agregar los RoleBindings adicionales que no sean el valor predeterminado administrator@vsphere.local. Por ejemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    run.tanzu.vmware.com/vmware-system-synced-from-supervisor: "yes"
  name: vmware-system-auth-sync-wcp:custom-ns:group:vsphere.local:administrators
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: sso:Administrators@vsphere.local
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    run.tanzu.vmware.com/vmware-system-synced-from-supervisor: "yes"
  name: vmware-system-auth-sync-wcp:custom-ns:group:SSODOMAIN.COM:testuser
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
```

```
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: sso:testuser@SSODOMAIN.COM
```

**Nota** En el campo `metadata.name`, la función de usuario se antepone con `vmware-system-auth-sync-` para todos los usuarios. Las entradas metadata.name y subjects.name requerirán modificaciones para todas las funciones no predeterminadas.

4 Aplique la configuración sync-cluster-edit-rolebinding.yaml para sincronizar los RoleBindings.

```
KUBECONFIG=ccc-cluster-kubeconfig kubectl apply -f sync-cluster-edit-rolebinding.yaml
```

# Usar la API v1alpha3 del clúster de Tanzu Kubernetes

En esta sección se proporciona contenido de referencia para aprovisionar un clúster de servicio TKG de tipo `TanzuKubernetesCluster` mediante la API v1alpha3, incluidos ejemplos con diversas configuraciones y personalizaciones para satisfacer sus necesidades

## API v1alpha3 del clúster de Tanzu Kubernetes

La API v1alpha3 permite aprovisionar un clúster de Tanzu Kubernetes mediante TKG en Supervisor. Consulte este tema para ver la documentación de la API de v1alpha3.

### API v1alpha3 del clúster de Tanzu Kubernetes

La especificación enumera todos los parámetros disponibles para aprovisionar una instancia de `TanzuKubernetesCluster` mediante la API v1alpha3.

**Importante** Un nombre de clave válido solo debe constar de caracteres alfanuméricos, un guion (como `key-name`), un guion bajo (como `KEY_NAME`) o un punto (como `key.name`). No puede utilizar el carácter de espacio en un nombre de clave.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: string
  namespace: string
  annotations:
    run.tanzu.vmware.com/resolve-os-image: os-name=string
spec:
  topology:
    controlPlane:
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
```

```
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
    nodePools:
    - name: string
      failureDomain: string
      labels: map[string]string
      taints:
        -  key: string
           value: string
           effect: string
           timeAdded: time
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
  settings:
    storage:
      classes: [string]
      defaultClass: string
    network:
      cni:
        name: string
      pods:
        cidrBlocks: [string]
      services:
        cidrBlocks: [string]
      serviceDomain: string
      proxy:
        httpProxy: string
        httpsProxy: string
        noProxy: [string]
      trust:
        additionalTrustedCAs:
          - name: string
            data: string
```

## API de clúster de Tanzu Kubernetes v1alpha3: anotada

La especificación anotada enumera todos los parámetros disponibles para aprovisionar un clúster de Tanzu Kubernetes mediante la API v1alpha3 con la documentación de cada campo.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
#valid config key must consist of alphanumeric characters, '-', '_' or '.'
#metadata defines cluster information
metadata:
  #name for this Tanzu Kubernetes cluster
  name: string
  #namespace vSphere Namespace where to provision this cluster
  namespace: string
  #Use annotation to provision non-default OS for the VM nodes
  #PhotonOS is the default OS; use "ubuntu" to specify Ubuntu TKR
  annotations:
    run.tanzu.vmware.com/resolve-os-image: os-name=string
#spec defines cluster configuration
spec:
  #topology describes the number, purpose, organization of nodes
  #and the resources allocated for each
  #nodes are grouped into pools based on their purpose
  #controlPlane is special kind of a node pool
  #nodePools is for groups of worker nodes
  #each node pool is homogeneous: its nodes have the same
  #resource allocation and use the same storage
  topology:
    #controlPlane defines the topology of the cluster
    #controller, including the number of nodes and
    #the resources allocated for each
    #control plane must have an odd number of nodes
    controlPlane:
      #replicas is the number of nodes in the pool
      #the control plane can have 1 or 3 nodes
      #NOTE: production deployments require 3 control plane nodes
      #defaults to 1 if nil (empty)
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the node pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #kubectl get virtualmachineclass
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #kubectl describe storageclasses
      storageClass: string
      #volumes is the optional set of PVCs
      #to create and attach to each control plane node
      volumes:
        #name of the PVC to be used as the suffix (node.name)
```

```
      - name: string
        #mountPath is the directory where the volume
        #device is mounted; takes the form /dir/path
        mountPath: string
        #capacity is the PVC capacity
        capacity:
          #storage to be used for the disk
          #volume; if not specified defaults to
          #spec.controlPlane.storageClass
          storage: size in GiB
    #tkr.reference.name is the TKR NAME
    #to be used by control plane nodes
    #format is v1.27.11---vmware.1-fips.1-tkg.2
    #currently all tkr.reference.name fields must match
    tkr:
      reference:
        name: string
    #nodeDrainTimeout is the total amount of time
    #the controller will spend draining a node
    #the default value is 0 which means the node is
    #drained without any time limit
    nodeDrainTimeout: string
  #nodePools is an array that describes a group of
  #worker nodes in the cluster with the same configuration
  nodePools:
  #name of the worker node pool
  #must be unique in the cluster
  - name: string
    #failureDomain is the name of a vSphere Zone
    #failureDomain is required for multi-zoned Supervisor
    #in a multi-zoned Supervisor, you will have 3 node pools
    #each refernecing a different failureDomain zone name
    #refer to the examples
    failureDomain: string
    #labels are an optional map of string keys and values
    #to organize and categorize objects
    #propagated to the created nodes
    labels: map[string]string
    #taints specifies optional taints to register the
    #Node API object with; user-defined taints are
    #propagated to the created nodes
    taints:
      #key is the taint key to be applied to a node
      -  key: string
      #value is the taint value corresponding to the key
         value: string
      #effect is the effect of the taint on pods
      #that do not tolerate the taint; valid effects are
      #NoSchedule, PreferNoSchedule, NoExecute
         effect: string
      #timeAdded is the time when the taint was added
      #only written by the system for NoExecute taints
         timeAdded: time
    #replicas is the number of nodes in the pool
    #worker nodePool can have from 0 to 150 nodes
```

```
      #value of nil means the field is not reconciled,
      #allowing external services like autoscalers
      #to choose the number of nodes for the nodePool
      #by default CAPI's MachineDeployment will pick 1
      #NOTE: a cluster provisioned with 0 worker nodes/nodepools
      #is not assigned any load balancer services
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #kubectl get virtualmachineclass
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #kubectl describe ns
      storageClass: string
      #volumes is the optional set of PVCs to create
      #and attach to each node for high-churn worker node
      #components such as the container runtime
      volumes:
        #name of this PVC to be used as the suffix (node.name)
        - name: string
          #mountPath is the directory where the volume
          #device is mounted; takes the form /dir/path
          mountPath: string
          #capacity is the PVC capacity
          capacity:
            #storage to be used for the disk
            #volume; if not specified defaults to
            #topology.nodePools[*].storageClass
            storage: size in GiB
      #tkr.reference.name points to the TKR NAME
      #to be used by spec.topology.nodePools[*] nodes
      #format is v1.27.11---vmware.1-fips.1-tkg.2
      #currently all tkr.reference.name fields must match
      tkr:
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
  #settings are optional runtime configurations
  #for the cluster, including persistent storage
  #for pods and node network customizations
  settings:
    #storage defines persistent volume (PV) storage entries
    #for container workloads; note that the storage used for
    #node disks is defined by topology.controlPlane.storageClass
    #and by spec.topology.nodePools[*].storageClass
```

```
    storage:
      #classes is a list of persistent volume (PV) storage
      #classes to expose for container workloads on the cluster
      #any class specified must be associated with the
      #vSphere Namespace where the cluster is provisioned
      #if omitted, all storage classes associated with the
      #namespace will be exposed in the cluster
      classes: [string]
      #defaultClass treats the named storage class as the default
      #for the cluster; because all namespaced storage classes
      #are exposed if specific classes are not named,
      #classes is not required to specify a defaultClass
      #many workloads, including TKG Extensions and Helm,
      #require a default storage class
      #if omitted, no default storage class is set
      defaultClass: string
    #netowrk defines custom networking for cluster workloads
    network:
      #cni identifies the CNI plugin for the cluster
      #use to override the default CNI set in the
      #tkgservicesonfiguration spec, or when customizing
      #network settings for the default CNI
      cni:
        #name is the name of the CNI plugin to use
        #supported values are antrea, calico, antrea-nsx-routed
        name: string
      #pods configures custom networks for pods
      #defaults to 192.168.0.0/16 if CNI is antrea or calico
      #defaults to empty if CNI is antrea-nsx-routed
      #custom subnet size must equal or exceed /24
      #use caution before seeting CIDR range other than /16
      #cannot overlap with Supervisor workload network
      pods:
        #cidrBlocks is an array of network ranges
        #multiple ranges may not be supported by all CNI plugins
        cidrBlocks: [string]
      #services configures custom network for services
      #defaults to 10.96.0.0/12
      #cannot overlap with Supervisor workload network
      services:
        #cidrBlocks is an array of network ranges
        #multiple ranges many not be supported by all CNI plugins
        cidrBlocks: [string]
      #serviceDomain specifies the service domain for the cluster
      #defaults to cluster.local
      serviceDomain: string
      #proxy configures proxy server to be used inside the cluster
      #if omitted no proxy is configured
      proxy:
        #httpProxy is the proxy URI for HTTP connections
        #to endpoints outside the cluster
        #takes form http://<user>:<pwd>@<ip>:<port>
        httpProxy: string
        #httpsProxy is the proxy URL for HTTPS connections
        #to endpoints outside the cluster
```

```
        #takes the frorm http://<user>:<pwd>@<ip>:<port>
        httpsProxy: string
        #noProxy is the list of destination domain names, domains,
        #IP addresses, and other network CIDRs to exclude from proxying
        #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
        noProxy: [string]
      #trust configures additional certificates for the cluster
      #if omitted no additional certificate is configured
      trust:
        #additionalTrustedCAs are additional trusted certificates
        #can be additional CAs or end certificates
        additionalTrustedCAs:
          #name is the name of the additional trusted certificate
          #must match the name used in the filename
          - name: string
            #data holds the contents of the additional trusted cert
            #PEM Public Certificate data as a base64-encoded string
            #such as LS0tLS1C...LS0tCg== where "..." is the
            #middle section of the long base64-encoded string
            data: string
```

# Ejemplo de v1alpha3: clúster de Tanzu Kubernetes predeterminado

Consulte el ejemplo de YAML para aprovisionar un clúster de Tanzu Kubernetes predeterminado mediante la API v1alpha3.

## Ejemplo de v1alpha3: clúster de Tanzu Kubernetes predeterminado

El ejemplo de YAML aprovisiona un clúster de Tanzu Kubernetes predeterminado mediante la API v1alpha3.

Este ejemplo muestra la configuración mínima requerida para aprovisionar un TKC. La configuración de red y almacenamiento predeterminada se utiliza y se excluye del YAML. La TKR a la que se hace referencia se utiliza para el plano de control y los nodos de trabajo.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-default
  namespace: tkg-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
```

```
    - replicas: 3
      name: worker
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
```

# Ejemplo de v1alpha3: TKC con almacenamiento predeterminado y volúmenes de nodos

Consulte el ejemplo de YAML para aprovisionar un clúster de Tanzu Kubernetes mediante la API v1alpha3 con una clase de almacenamiento predeterminada y una configuración personalizada para los volúmenes de nodos.

## Ejemplo de v1alpha3: TKC con volúmenes de nodo y almacenamiento predeterminado

El ejemplo de YAML aprovisiona un clúster de Tanzu Kubernetes personalizado mediante la API v1alpha3.

Observe las siguientes personalizaciones opcionales en este ejemplo. Consulte la API v1alpha3 del clúster de Tanzu Kubernetes para obtener más información.

- El clúster se aprovisiona con una clase de almacenamiento predeterminada, que requieren algunas herramientas, como las cargas de trabajo implementadas por los paquetes de Helm y Tanzu
- Los volúmenes de nodo de trabajo se declaran para componentes de renovación alta, como `containerd` y `kubelet`

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-custom-storage
  namespace: tkg-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
    - replicas: 3
      name: worker-np
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
```

```
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 50Gi
  settings:
    storage:
      defaultClass: tkg-storage-policy
```

# Ejemplo de v1alpha3: TKC con red personalizada

Consulte el ejemplo de YAML para aprovisionar un clúster de Tanzu Kubernetes mediante la API v1alpha3 con la configuración de red personalizada.

## Ejemplo de v1alpha3: TKC con configuración personalizada

La red se personaliza de la siguiente manera. Consulte la API v1alpha3 del clúster de Tanzu Kubernetes para obtener más información.

- La CNI de Calico se utiliza en lugar de la Antrea predeterminada
- Se utilizan subredes no predeterminadas para pods y servicios
- Se declaran un servidor proxy y certificados TLS

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-custom-network
  namespace: tkg2-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
    - name: worker
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
```

```
        capacity:
          storage: 50Gi
  settings:
    storage:
      defaultClass: tkg-storage-policy
    network:
      cni:
        name: calico
      services:
        cidrBlocks: ["172.16.0.0/16"]
      pods:
        cidrBlocks: ["192.168.0.0/16"]
      serviceDomain: cluster.local
      proxy:
        httpProxy: http://<user>:<pwd>@<ip>:<port>
        httpsProxy: http://<user>:<pwd>@<ip>:<port>
        noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
      trust:
        additionalTrustedCAs:
          - name: CompanyInternalCA-1
            data: LS0tLS1C...LS0tCg==
          - name: CompanyInternalCA-2
            data: MTLtMT1C...MT0tPg==
```

## Consideraciones para personalizar la red de pods de TKC

El valor predeterminado de la configuración de la especificación del clúster `spec.settings.network.pods.cidrBlocks` es 192.168.0.0/16.

Si personaliza, el tamaño mínimo del bloque CIDR de los pods es /24. Sin embargo; tenga cuidado al restringir la máscara de subred de `pods.cidrBlocks` más allá de /16.

TKG asigna a cada nodo de clúster una subred /24 creada desde `pods.cidrBlocks`. Esta asignación se determina mediante el parámetro Administrador de controladoras de Kubernetes > NodeIPAMController denominado NodeCIDRMaskSize que establece el tamaño de la máscara de subred para el CIDR de nodo en el clúster. La máscara de subred del nodo predeterminada es /24 para IPv4.

Debido a que cada nodo de un clúster obtiene una subred /24 de `pods.cidrBlocks`, puede quedarse sin direcciones IP de nodo si utiliza un tamaño de máscara de subred que sea demasiado restrictivo para el clúster que está aprovisionando.

Los siguientes límites de nodos se aplican a un clúster de Tanzu Kubernetes aprovisionado con la CNI de Antrea o Calico.

/16 == 150 nodos máx. (por ConfigMax)

/17 == 128 nodos máx.

/18 == 64 nodos como máximo

/19 == 32 nodos como máximo

/20 == 16 nodos como máximo

/21 == 8 nodos como máximo

/22 == 4 nodos como máximo

/23 == 2 nodos como máximo

/24 == 1 nodo máx.

## Ejemplo de v1alpha3: TKC con Ubuntu TKR

Consulte el ejemplo de YAML que se proporciona aquí para aprovisionar un clúster de Tanzu Kubernetes que utiliza el sistema operativo Ubuntu para los nodos de clúster. Este tipo de clúster se puede utilizar para cargas de trabajo de vGPU.

### Ejemplo de v1alpha3: TKC con Ubuntu TKR

De forma predeterminada, la edición PhotonOS de la TKR con nombre se utiliza para los nodos de clúster de TKG. Si el TKR al que se hace referencia admite el formato OSImage y tiene una edición del sistema operativo Ubuntu disponible, utilice la anotación `run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu` para especificar la edición del sistema operativo Ubuntu del TKR. Para obtener más información sobre los próximos pasos, consulte Formato TKr Imagen de sistema operativo.

El TKR de Ubuntu es necesario para las cargas de trabajo de IA/ML. Cada grupo de nodos de trabajo tiene un volumen independiente para el tiempo de ejecución en contenedor y kubelet, cada uno con una capacidad de 70 GiB. Se recomienda proporcionar un volumen independiente de este tamaño para las cargas de trabajo de AI/ML basadas en contenedores.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-ubuntu-gpu
  namespace: tkg-cluster-ns
  annotations:
    run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
spec:
   topology:
     controlPlane:
       replicas: 3
       storageClass: tkg-storage-policy
       vmClass: guaranteed-large
       tkr:
         reference:
           name: v1.25.7---vmware.3-fips.1-tkg.1
     nodePools:
     - name: nodepool-a100-primary
       replicas: 3
       storageClass: tkg-storage-policy
       vmClass: vgpu-a100
       tkr:
         reference:
           name: v1.25.7---vmware.3-fips.1-tkg.1
       volumes:
```

```
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 70Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 70Gi
    - name: nodepool-a100-secondary
      replicas: 3
      storageClass: tkg-storage-policy
      vmClass: vgpu-a100
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 70Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 70Gi
  settings:
    storage:
      defaultClass: tkg-storage-policy
    network:
      cni:
       name: antrea
      services:
       cidrBlocks: ["198.51.100.0/12"]
      pods:
       cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: cluster.local
```

# Ejemplo de v1alpha3: TKC entre zonas de vSphere

Consulte el ejemplo de YAML para aprovisionar un clúster de Tanzu Kubernetes entre zonas de vSphere mediante la API v1alpha3.

## Zonas de vSphere y dominios de errores

Las zonas de vSphere permiten crear clústeres de TKG de alta disponibilidad en Supervisor. Para aprovisionar un clúster de TKG en zonas de vSphere, debe proporcionar el dominio de errores para cada grupo de nodos.

Cada dominio de errores se asigna a una zona de vSphere que se asociará con un clúster de vSphere. Los dominios de errores, también conocidos como "dominios de errores de vSphere", son definidos y administrados por el administrador de vSphere al crear zonas de vSphere. El perfil de almacenamiento que se utiliza para el clúster de TKG debe configurarse como `zonal`. Consulte Crear una directiva de almacenamiento de vSphere para clústeres de Servicio TKG.

Cuando se implementan pods con réplicas en un clúster de TKG en Supervisor, las instancias del pod se distribuyen automáticamente entre las zonas de vSphere. No es necesario proporcionar detalles de la zona al implementar el pod en el clúster de TKG.

Para comprobar la disponibilidad de las zonas de vSphere en el entorno de TKG, ejecute los siguientes comandos desde el espacio de nombres de vSphere en el que aprovisionará el clúster de TKG:

```
kubectl get vspherezones
```

```
kubectl get availabilityzones
```

Ambos comandos están disponibles para los usuarios de `system:authenticated`. Las zonas de vSphere son recursos del ámbito de Supervisor, por lo que no necesita especificar un espacio de nombres.

## Ejemplo de v1alpha3: TKC entre zonas de vSphere

En el ejemplo de YAML, se aprovisiona un clúster de TKG en zonas de vSphere.

En este ejemplo, se especifica la zona de vSphere en el parámetro `failureDomain` para cada grupo de nodos. El valor de este parámetro es el nombre de la zona de vSphere.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-zoned
  namespace: tkg-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg2-storage-policy-zonal
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
      - name: nodepool-a01
        replicas: 3
        vmClass: guaranteed-medium
        storageClass: tkg-storage-policy-zonal
        failureDomain: az1
      - name: nodepool-a02
        replicas: 3
        vmClass: guaranteed-medium
        storageClass: tkg-storage-policy-zonal
        failureDomain: az2
      - name: nodepool-a03
        replicas: 3
        vmClass: guaranteed-medium
        storageClass: tkg-storage-policy-zonal
        failureDomain: az3
```

```
  settings:
    storage:
      defaultClass: tkg-storage-policy-zonal
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: cluster.local
```

## Ejemplo de v1alpha3: TKC con red de pods enrutables

Puede crear un clúster de Tanzu Kubernetes con redes de pods enrutables si configura una red de espacio de nombres enrutable en Supervisor y especifica `antrea-nsx-routed` como CNI para el clúster.

### Acerca de las redes de pods enrutables

Cuando se aprovisiona un clúster de Tanzu Kubernetes mediante los complementos de CNI `antrea` o `calico`, el sistema crea la red de pods predeterminada `192.168.0.0/16`. Esta subred es un espacio de direcciones privadas que solo es único dentro del clúster y que no se puede enrutar en la red.

La API v1alpha3 de TKG admite redes de pods enrutables mediante el complemento de CNI de `antrea-nsx-routed`. Esta interfaz de red es un complemento de Antrea personalizado configurado para admitir redes de pods enrutables para clústeres de TKG. En la especificación de clúster, el campo de bloques CIDR de pods debe ser explícitamente nulo para que la administración de direcciones IP (IPAM) se controle mediante Supervisor. Consulte el ejemplo a continuación.

Habilitar redes de pods enrutables permite que los pods se direccionen directamente desde un cliente externo al clúster. Además, las direcciones IP de los pods se conservan para que los servidores y los servicios de red externos puedan identificar los pods de origen y aplicar directivas basadas en direcciones IP. Patrones de tráfico compatibles, incluidos los siguientes:

- Se permite el tráfico entre un pod de clúster de TKG y un pod de vSphere en el mismo espacio de nombres de vSphere.
- El tráfico se descarta entre un pod de clúster de TKG y un pod de vSphere en diferentes espacios de nombres de vSphere.
- Los nodos de plano de control de Supervisor pueden acceder a los pods del clúster de TKG.
- Los pods del clúster de TKG pueden comunicarse con la red externa.
- La red externa no puede acceder a los pods del clúster de TKG. Las reglas de aislamiento de firewall distribuido (DFW) descartan el tráfico en los nodos de clúster.

## Crear una red de pods enrutables: configuración de Supervisor

Para crear una red de pods enrutables se requiere la configuración en el Supervisor y en el clúster de TKG.

**Nota** El Supervisor debe configurarse con NSX para usar redes de pods enrutables. No se pueden utilizar pods enrutables con redes de VDS.

Para configurar una red de pods enrutables en Supervisor:

1 Cree un nuevo espacio de nombres de vSphere.

   Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

2 Seleccione la opción de casilla de verificación para **Anular configuración de red de supervisor**.

   Consulte Anular la configuración de la red de cargas de trabajo para un espacio de nombres de vSphere para obtener instrucciones.

3 Configure la red de pods enrutables de la siguiente manera.

| Campo | Descripción |
| --- | --- |
| **Modo NAT** | Anule la selección de esta opción para deshabilitar la traducción de direcciones de red (Network Address Translation, NAT), ya que está utilizando una subred enrutable. |
| **CIDR de red de espacio de nombres** | El CIDR de red de espacio de nombres es una subred que funciona como un grupo de direcciones IP para el espacio de nombres de vSphere. El prefijo de subred de espacio de nombres describe el tamaño de cualquier bloque CIDR posterior separado de ese grupo de direcciones IP.<br>Rellene este campo con una subred IP enrutable con el formato Dirección IP/Bits (p. ej., 10.0.0.6/16). NCP creará uno o varios grupos de direcciones IP a partir de los bloques de IP especificados para la red.<br>Como mínimo, debe especificar un tamaño de subred /23. Por ejemplo, si especifica una subred enrutable /23 con un prefijo de subred /28, obtendrá 32 subredes que deberían ser suficientes para un clúster de 6 nodos. Una subred /24 con un prefijo /28 solo obtendrá 2 subredes, que no son suficientes. |
| **Prefijo de subred de espacio de nombres** | El prefijo de subred de espacio de nombres describe el tamaño de cualquier bloque CIDR posterior separado del grupo de direcciones IP de la red de espacio de nombres.<br>Especifique un prefijo de subred con el formato /28, por ejemplo. |

4 Haga clic en **Crear** para crear la red de pods enrutables.

### Crear una red de pods enrutables: configuración de un clúster de TKG

El siguiente ejemplo de YAML muestra cómo configurar un clúster con una red de pods enrutable.

La especificación del clúster declara `antrea-nsx-routed` como la CNI para habilitar las redes de pods enrutables. Si se especifica `antrea-nsx-routed`, se producirá un error en el aprovisionamiento del clúster si no se utilizan redes de NSX-T.

Cuando se especifica la CNI como `antrea-nsx-routed`, el campo `pods.cidrBlock` debe estar vacío.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-routable-pods
  namespace: tkg-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: tkg-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
  settings:
    storage:
      defaultClass: tkg-storage-policy
    network:
      #antrea-nsx-routed is the required CNI
      cni:
        name: antrea-nsx-routed
      services:
        cidrBlocks: ["10.97.0.0/24"]
      #pods.cidrBlocks must be null (empty)
      pods:
        cidrBlocks:
      serviceDomain: cluster.local
```

## Ejemplo de v1alpha3: TKC con certificados de CA de confianza adicionales para SSL/TLS

Consulte el ejemplo de YAML para aprovisionar un clúster de Tanzu Kubernetes mediante la API v1alpha3 con certificados de CA de confianza adicionales para SSL/TLS.

## Ejemplo de v1alpha3: TKC con certificados de CA de confianza adicionales

El clúster se personaliza de la siguiente manera. Consulte la API v1alpha3 del clúster de Tanzu Kubernetes para obtener más información.

- Los certificados de CA de confianza adicionales se declaran en la sección `network.trust.additionalTrustedCAs` de la especificación del clúster
- El campo `additionalTrustedCAs` es una matriz de pares nombre-valor:
  - El campo `name` es una cadena definida por el usuario
  - El valor `data` es el contenido del certificado de CA en formato PEM con codificación base64

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc-additional-trusted-cas
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkgs-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
    nodePools:
    - name: worker
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: tkgs-storage-policy
      tkr:
        reference:
          name: v1.25.7---vmware.3-fips.1-tkg.1
  settings:
    storage:
      defaultClass: tkgs-storage-policy
    network:
      trust:
        additionalTrustedCAs:
          - name: CompanyInternalCA-1
            data: LS0tLS1C...LS0tCg==
          - name: CompanyInternalCA-2
            data: MTLtMT1C...MT0tPg==
```

## Procedimiento: Nuevo clúster

Complete el siguiente procedimiento para incluir uno o varios certificados de CA de confianza adicionales en un nuevo clúster de TKGS.

1 Rellene el campo `additionalTrustedCAs` con el nombre y el valor de datos de uno o varios certificados de CA.

2 Aprovisione el clúster como lo haría normalmente.

   Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

3 Cuando el clúster se aprovisiona correctamente, los certificados de CA que agregó son de confianza para el clúster.

## Procedimiento: Clúster existente

Complete el siguiente procedimiento para agregar uno o varios certificados de CA de confianza adicionales a un clúster existente.

1 Compruebe que configuró kubectl editing.

   Consulte Configurar un editor de texto para Kubectl.

2 Edite la especificación del clúster.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-name
```

3 Agregue la sección `network.trust.additionalTrustedCAs` a la especificación.

4 Rellene el campo `additionalTrustedCAs` con el nombre y el valor de datos de uno o varios certificados de CA.

5 Guarde los cambios en el editor de texto y compruebe que kubectl los haya registrado.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-name
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-name edited
```

6 Cuando se inicia una actualización gradual para el clúster, se agregan los certificados de CA de confianza adicionales.

   Consulte Información sobre el modelo de actualización gradual para clústeres de Servicio TKG.

## Comprobar los certificados de CA de confianza adicionales

Los certificados de CA de confianza adicionales agregados al clúster se incluyen en el archivo kubeconfig del clúster.

## Solucionar los problemas de certificados de CA de confianza adicionales

Consulte Solucionar errores con certificados de CA de confianza adicionales.

## Caso práctico

El caso práctico más común es para agregar una CA de confianza adicional para conectarse a un registro de contenedor. Consulte Integrar clústeres de Servicio TKG con un registro de contenedor privado.

# Operar clústeres del servicio de TKG

8

En esta sección, se tratan temas para la operación de clústeres del servicio de TKG.

Lea los siguientes temas a continuación:



## Configurar un editor de texto para Kubectl

Para operar y mantener los clústeres de TKG, configure un editor de texto predeterminado para kubectl.

## Usar el comando kubectl edit

Después de aprovisionar un clúster de TKG, debe usarlo y mantenerlo. Entre las tareas típicas se incluye escalar nodos del clúster y actualizar la versión de TKR. Para realizar estas tareas, actualice el manifiesto del clúster mediante el comando kubectl edit.

El comando `kubectl edit` *`CLUSTER-KIND`*/*`CLUSTER-NAME`* abre el manifiesto del clúster en el editor de texto definido por la variable de entorno KUBE_EDITOR o EDITOR. Al guardar los cambios del manifiesto, `kubectl` informa que las modificaciones se registraron correctamente, y el clúster se actualiza con los cambios.

Por ejemplo:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkg-cluster-1 edited
```

Para cancelar los cambios, cierre el editor sin guardar.

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
Edit cancelled, no changes made.
```

## Configurar kubectl edit

Para usar el comando `kubectl edit`, se establece la variable de entorno EDITOR en Linux. De lo contrario, cree una variable de entorno KUBE_EDITOR y especifique el editor de texto preferido como el valor de la variable. Anexe la marca de inspección ( `-w`) para que kubectl sepa cuándo se confirmaron (se guardaron) los cambios.

Consulte las instrucciones específicas de su sistema operativo.

**Linux**

Para configurar `kubectl edit` en Linux (Ubuntu, por ejemplo), la línea de comandos predeterminada `EDITOR` es VIM. Si es así, no se necesita ninguna otra acción para usar el comando `kubectl edit`.

Si desea utilizar otro editor de texto, cree una variable de entorno llamada `KUBE_EDITOR` con el valor establecido en la ruta de acceso del editor de texto elegido.

**Mac OS**

Para configurar `kubectl edit` en Mac OS, cree una variable de entorno llamada `KUBE_EDITOR` con el valor establecido en la ruta de acceso del editor de texto elegido. Anexe la marca de espera ( `--wait` o el acceso directo `-w`) al valor para que el editor sepa cuándo se confirmaron (se guardaron) los cambios.

Por ejemplo, la siguiente adición a `.bash_profile` establece Sublime como el editor de texto predeterminado para kubectl e incluye la marca de espera para que el editor sepa cuándo se guardaron los cambios.

```
export KUBE_EDITOR="/Applications/Sublime.app/Contents/SharedSupport/bin/subl -w"
```

**Windows**

Para configurar `kubectl edit` en Windows, cree una variable de entorno del sistema llamada `KUBE_EDITOR` con el valor establecido en la ruta de acceso del editor de texto elegido. Anexe la marca de inspección ( `-w`) al valor.

Por ejemplo, la siguiente variable de entorno establece el código de Visual Studio como el editor de texto predeterminado para kubectl e incluye la marca de inspección para que Kubernetes sepa cuándo se guardaron los cambios:

```
KUBE_EDITOR=code -w
```

Para configurar Sublime como editor kubectl en Windows, anexe el directorio del programa Sublime a la ruta del sistema y cree una variable del sistema para el ejecutable de Sublime. Por ejemplo:

Anexar ruta de acceso del sistema:

```
C:\Program Files\Sublime Text 3\
```

Nombre y valor de la variable del sistema:

```
KUBE_EDITOR=sublime_text.exe -w
```

# Escalar manualmente un clúster mediante Kubectl

Puede escalar un clúster de servicio TKG horizontalmente cambiando el número de nodos, o bien verticalmente cambiando la clase de máquina virtual que aloja los nodos. También puede escalar volúmenes asociados a nodos del clúster.

## Operaciones de escala manual admitidas

En la tabla se enumeran las operaciones de escalado admitidas para clústeres de TKG.

**Tabla 8-1. Operaciones de escalado admitidas para clústeres de TKGS**

| Nodo | Expansión horizontal | Reducción horizontal | Ampliación vertical | Escala de volumen |
|---|---|---|---|---|
| Plano de control | Sí | No | Sí | Sí* |
| Trabajador | Sí | Sí | Sí | Sí |

Tenga en cuenta las siguientes consideraciones:

- La cantidad de nodos de plano de control debe ser impar, ya sea 1 o 3. Se admite el escalado horizontal del plano de control, pero no se admite la reducción horizontal del plano de control. Consulte Ampliar horizontalmente el plano de control.
- Al escalar verticalmente un nodo de clúster, es posible que las cargas de trabajo ya no puedan ejecutarse en el nodo por falta de recursos disponibles. Por lo tanto, el escalado horizontal es generalmente el método preferido. Consulte Escalar horizontalmente los nodos de trabajo.
- Las clases de máquina virtual no son inmutables. Si se escala horizontalmente un clúster de TKG después de editar una clase de máquina virtual utilizada por ese clúster, los nuevos nodos del clúster utilizan la definición de clase actualizada, pero los nodos del clúster existentes siguen usando la definición de clase inicial, lo que provoca un error de coincidencia. Consulte Acerca de las clases de máquina virtual.
- Los volúmenes de nodos de trabajo se pueden cambiar después del aprovisionamiento. Del mismo modo, los nodos del plano de control se pueden cambiar siempre que utilice vSphere 8 U3 o una versión posterior y un TKr compatible. Consulte Escalar volúmenes de nodos de clúster.

## Requisito previo para la ampliación: configurar la edición de Kubectl

Para escalar un clúster de TKG, actualiza el manifiesto del clúster usando el comando `kubectl edit CLUSTER-KIND/CLUSTER-NAME`. Al guardar los cambios en el manifiesto, el clúster se actualiza con los cambios. Consulte Configurar un editor de texto para Kubectl.

Por ejemplo:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkg-cluster-1 edited
```

Para cancelar los cambios, cierre el editor sin guardar.

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
Edit cancelled, no changes made.
```

## Ampliar horizontalmente el plano de control

Para escalar horizontalmente un clúster de TKG, aumente el número de nodos del plano de control de 1 a 3.

**Nota** Los clústeres de producción requieren 3 nodos del plano de control.

1 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=SUPERVISOR-IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere en el que se ejecuta el clúster de TKG.

```
kubectl config use-context tkg-cluster-ns
```

3 Enumere los clústeres de Kubernetes que se ejecutan en espacio de nombres de vSphere.

Utilice la sintaxis siguiente:

```
kubectl get CLUSTER-KIND -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1beta1:

```
kubectl get cluster -n tkg-cluster-ns
```

4 Obtenga la cantidad de nodos que se ejecutan en el clúster de destino.

Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

El clúster de TKG tiene 1 nodo de plano de control y 3 nodos de trabajo.

```
NAMESPACE           NAME            CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns      tkg-cluster-1   1               3        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

5 Cargue el manifiesto del clúster para editarlo ejecutando el comando `kubectl edit`.

Para un clúster de API v1alpha3:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
```

Para un clúster de API v1beta1:

```
kubectl edit cluster/tkg-cluster-1
```

El manifiesto del clúster se abrirá en el editor de texto que definan las variables de entorno KUBE_EDITOR o EDITOR.

6 Aumente el número de nodos del plano de control de 1 a 3 en la sección `spec.topology.controlPlane.replicas` del manifiesto.

Para un clúster de API v1alpha3:

```
...
spec:
  topology:
    controlPlane:
      replicas: 1
...
```

```
...
spec:
  topology:
    controlPlane:
      replicas: 3
...
```

Para un clúster de API v1beta1:

```
...
spec:
  ...
  topology:
    class: tanzukubernetescluster
    controlPlane:
      metadata: {}
      replicas: 1
    variables:
...
```

```
...
spec:
  ...
  topology:
    class: tanzukubernetescluster
    controlPlane:
      metadata: {}
      replicas: 3
    variables:
...
```

7 Guarde el archivo en el editor de texto para aplicar los cambios. (Para cancelar, cierre el editor sin guardar).

   Cuando guarde el manifiesto, kubectl aplicará los cambios al clúster. En segundo plano, la instancia de servicio de máquina virtual de Supervisor aprovisiona el nuevo nodo de plano de control.

8 Compruebe que se agreguen los nodos nuevos.

   Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

El plano de control que se amplió horizontalmente ahora tiene 3 nodos.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns        tkg-cluster-1    3               3        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

## Escalar horizontalmente los nodos de trabajo

Para escalar horizontalmente un clúster de TKG, aumente el número de nodos de trabajo.

1 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere en el que se ejecuta el clúster de TKG.

```
kubectl config use-context tkg-cluster-ns
```

3 Enumere los clústeres de Kubernetes que se ejecutan en espacio de nombres de vSphere.

Utilice la sintaxis siguiente:

```
kubectl get CLUSTER-KIND -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1alph3:

```
kubectl get tanzukubernetescluster -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1beta1:

```
kubectl get cluster -n tkg-cluster-ns
```

4 Obtenga la cantidad de nodos que se ejecutan en el clúster de destino.

Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

Por ejemplo, el siguiente clúster tiene 3 nodo de plano de control y 3 nodos de trabajo.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns        tkg-cluster-1    3               3        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

5 Cargue el manifiesto del clúster para editarlo ejecutando el comando `kubectl edit`.

Para un clúster de API v1alpha3:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
```

Para un clúster de API v1beta1:

```
kubectl edit cluster/tkg-cluster-1
```

El manifiesto del clúster se abrirá en el editor de texto que definan las variables de entorno KUBE_EDITOR o EDITOR.

6 Aumente el número de nodos de trabajo editando el valor `spec.topology.nodePools.NAME.replicas` del grupo de nodos de trabajo de destino.

Para un clúster de API v1alpha3:

```
...
spec:
  topology:
    ...
    nodePools:
    - name: worker-1
      replicas: 3
...
```

```
...
spec:
  topology:
    ...
    nodePools:
    - name: worker-1
      replicas: 4
...
```

Para un clúster de API v1beta1:

```
...
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  ...
spec:
  ...
  topology:
    ...
    class: tanzukubernetescluster
    controlPlane:
```

```
    ...
    workers:
      machineDeployments:
      - class: node-pool
        metadata: {}
        name: node-pool-1
        replicas: 3
...
```

```
...
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  ...
spec:
  ...
  topology:
    ...
    class: tanzukubernetescluster
    controlPlane:
    ...
    workers:
      machineDeployments:
      - class: node-pool
        metadata: {}
        name: node-pool-1
        replicas: 4
...
```

7 Para aplicar los cambios, guarde el archivo en el editor de texto. Para cancelar los cambios, cierre el editor sin guardar.

Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, el servicio de máquina virtual del Supervisor aprovisiona el nuevo nodo de trabajo.

8 Compruebe que se agreguen los nodos nuevos.

Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

Después de ampliar, el clúster tiene 4 nodos de trabajo.

```

NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns        tkg-cluster      3               4        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

## Reducir los nodos de trabajo

Para reducir horizontalmente un clúster de TKG, disminuya el número de nodos de trabajo.

1 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere en el que se ejecuta el clúster de TKG.

```
kubectl config use-context tkg-cluster-ns
```

3 Enumere los clústeres de Kubernetes que se ejecutan en espacio de nombres de vSphere.

Utilice la sintaxis siguiente:

```
kubectl get CLUSTER-KIND -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1alph3:

```
kubectl get tanzukubernetescluster -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1beta1:

```
kubectl get cluster -n tkg-cluster-ns
```

4 Obtenga la cantidad de nodos que se ejecutan en el clúster de destino.

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

5 Obtenga la cantidad de nodos que se ejecutan en el clúster de destino.

Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

Por ejemplo, el siguiente clúster tiene 3 nodo de plano de control y 4 nodos de trabajo.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns        tkg-cluster      3               4        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

6 Cargue el manifiesto del clúster para editarlo ejecutando el comando `kubectl edit`.

Para un clúster de API v1alpha3:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
```

Para un clúster de API v1beta1:

```
kubectl edit cluster/tkg-cluster-1
```

El manifiesto del clúster se abrirá en el editor de texto que definan las variables de entorno KUBE_EDITOR o EDITOR.

7 Reduzca el número de nodos de trabajo editando el valor `spec.topology.nodePools.NAME.replicas` del grupo de nodos de trabajo de destino.

Para un clúster de API v1alpha3:

```
...
spec:
  topology:
    ...
    nodePools:
    - name: worker-1
      replicas: 4
...
```

```
...
spec:
  topology:
    ...
    nodePools:
    - name: worker-1
      replicas: 3
...
```

Para un clúster de API v1beta1:

```
...
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  ...
spec:
  ...
  topology:
    ...
    class: tanzukubernetescluster
    controlPlane:
    ...
    workers:
      machineDeployments:
      - class: node-pool
```

```
        metadata: {}
        name: node-pool-1
        replicas: 4
...
```

```
...
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  ...
spec:
  ...
  topology:
    ...
    class: tanzukubernetescluster
    controlPlane:
    ...
    workers:
      machineDeployments:
      - class: node-pool
        metadata: {}
        name: node-pool-1
        replicas: 3
...
```

8 Para aplicar los cambios, guarde el archivo en el editor de texto. Para cancelar los cambios, cierre el editor sin guardar.

Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, el servicio de máquina virtual del Supervisor aprovisiona el nuevo nodo de trabajo.

9 Compruebe que se hayan agregado los nodos de trabajo.

Para un clúster de API v1alpha3:

```
kubectl get tanzukubernetescluster tkg-cluster-1
```

Después de reducir, el clúster tiene 3 nodos de trabajo.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR NAME
AGE     READY
tkg-cluster-ns        tkg-cluster-1    3               3        v1.24.9---vmware.1-tkg.4
5d12h   True
```

Para un clúster de API v1beta1:

```
kubectl get cluster tkg-cluster-1
```

## Ampliar un clúster verticalmente

TKG en Supervisor admite el escalado vertical para los nodos de trabajo y de plano de control del clúster. Para escalar verticalmente un clúster de TKG, cambie la Uso de clases de máquinas virtuales con clústeres de Servicio TKG que se utiliza para los nodos del clúster. La clase de máquina virtual que use debe estar enlazada al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG.

TKG en Supervisor admite el escalado vertical a través del mecanismo de actualización gradual integrado en el sistema. Si cambia la definición de `VirtualMachineClass`, el servicio implementa gradualmente los nodos nuevos con esa nueva clase y reduce la velocidad de los nodos antiguos. Consulte Capítulo 9 Actualizar clústeres de servicio TKG.

1 Inicie sesión en Supervisor.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere en el que se ejecuta el clúster de TKG.

```
kubectl config use-context tkg-cluster-ns
```

3 Enumere los clústeres de Kubernetes que se ejecutan en espacio de nombres de vSphere.

Utilice la sintaxis siguiente:

```
kubectl get CLUSTER-KIND -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1alph3:

```
kubectl get tanzukubernetescluster -n tkg-cluster-ns
```

Por ejemplo, para un clúster de API v1beta1:

```
kubectl get cluster -n tkg-cluster-ns
```

4 Describa el clúster de TKG de destino y compruebe la clase de máquina virtual.

Para un clúster de API v1alpha3:

```
kubectl describe tanzukubernetescluster tkg-cluster-1
```

Por ejemplo, el siguiente clúster utiliza la clase de máquina virtual best-effort-medium.

```
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: best-effort-medium
      ...
    nodePools:
```

```
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: best-effort-medium
      ...
```

Para un clúster de API v1beta1:

```
kubectl describe cluster tkg-cluster-1
```

Por ejemplo, el siguiente clúster utiliza la clase de máquina virtual best-effort-medium.

```
...
Topology:
      ...
    Variables:
      ...
      Name:   vmClass
      Value:  best-effort-medium
    ...
```

**Nota** Para los clústeres de API v1beta1, vmClass se establece globalmente como una sola variable de forma predeterminada. Puede anular esta configuración y utilizar una clase de máquina virtual diferente para los nodos del plano de control y los nodos de trabajo. Consulte vmClassen la referencia de la API.

5 Enumere y describa las clases de máquinas virtuales disponibles.

```
kubectl get virtualmachineclass
```

```
kubectl describe virtualmachineclass
```

**Nota** La clase de máquina virtual debe estar enlazada al espacio de nombres de vSphere. Consulte Uso de clases de máquinas virtuales con clústeres de Servicio TKG.

6 Abra para editar el manifiesto del clúster de destino.

Para un clúster de API v1alpha3:

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
```

Para un clúster de API v1beta1:

```
kubectl edit cluster/tkg-cluster-1
```

El manifiesto del clúster se abrirá en el editor de texto que definan las variables de entorno KUBE_EDITOR o EDITOR.

7 Edite el manifiesto cambiando la clase de máquina virtual.

Para un clúster de API v1alpha3, cambie la clase de máquina virtual del plano de control a `guaranteed-medium` y la clase de máquina virtual para los nodos de trabajo a `guaranteed-large`.

```
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      ...
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      ...
```

Para un clúster de API v1beta, cambie la clase de máquina virtual a `guaranteed-large`.

```
...
Topology:
      ...
    Variables:
      ...
      Name:   vmClass
      Value:  guaranteed-large
    ...
```

8 Para aplicar los cambios, guarde el archivo en el editor de texto. Para cancelar los cambios, cierre el editor sin guardar.

   Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, el TKG en Supervisor realiza una actualización gradual del clúster de TKG.

9 Compruebe que el clúster de TKG esté actualizado con la nueva clase de máquina virtual.

   Para un clúster de API v1alpha3:

```
kubectl describe tanzukubernetescluster tkg-cluster-1
```

   Para un clúster de API v1beta1:

```
kubectl describe cluster tkg-cluster-1
```

## Escalar volúmenes de nodos de clúster

En la especificación del clúster de TKG para los nodos, de forma opcional, puede declarar uno o varios volúmenes persistentes para el nodo. La declaración de un volumen de nodos es útil para los componentes de alta rotación como el tiempo de ejecución del contenedor y kubelet en los nodos de trabajo.

Si desea agregar o cambiar uno o más volúmenes de nodos después de la creación del clúster, tenga en cuenta las siguientes consideraciones:

| Nodo de volumen | Descripción |
| --- | --- |
| Se permiten cambios en el volumen del nodo de trabajo | Después de aprovisionar un clúster de TKG, puede agregar o actualizar volúmenes de nodo de trabajo. Cuando se inicia una actualización gradual, el clúster se actualiza con el volumen nuevo o cambiado.<br><br>**Advertencia** Si escala el nodo de trabajo con un volumen nuevo o modificado, los datos del volumen actual se eliminan durante la actualización gradual. Consulte la explicación que aparece a continuación.<br><br>Un volumen declarado para un nodo de clúster de TKG se considera efímero. Un clúster de TKG utiliza una notificación de volumen persistente (PVC) en el espacio de nombres de vSphere para que la capacidad del volumen se cuente para la cuota de almacenamiento del clúster de TKG. Si aumenta la capacidad de un volumen TKC, la API del clúster de Kubernetes (CAPI) implementará nuevos trabajos con una nueva PVC. TKG no realiza ninguna migración de datos en este caso, pero Kubernetes (re)programará los pods de carga de trabajo según corresponda. |
| Se permiten cambios en el volumen de nodo de plano de control si utiliza vSphere 8 U3 o una versión posterior | Si utiliza vSphere 8 U3 o una versión posterior y una versión de Tanzu Kubernetes compatible, puede agregar o actualizar un volumen de nodo de plano de control después de que se haya aprovisionado un clúster de servicio TKG.<br>Si no utiliza vSphere 8 U3 o una versión posterior, la API de clúster de Kubernetes (CAPI) prohíbe los cambios posteriores a la creación en `spec.toplogy.controlPlane.volumes`.<br>Si intenta agregar o cambiar un volumen del plano de control después de la creación del clúster, se rechaza la solicitud y se recibe el mensaje de error "No se permiten las actualizaciones de los campos inmutables". |

A continuación se muestra una especificación de clúster extraída basada en la API v1alpha3 con volúmenes de nodo declarados. Consulte el ejemplo de clúster de TKG completo del que se toma este extracto según sea necesario: Ejemplo de v1alpha3: TKC con almacenamiento predeterminado y volúmenes de nodos. Para ver un ejemplo de clúster de API v1beta1, consulte Ejemplo de v1beta1: clúster personalizado basado en la ClusterClass predeterminada.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
...
spec:
   topology:
     controlPlane:
       replicas: 3
       storageClass: tkg-storage-policy
       vmClass: guaranteed-medium
       tkr:
         reference:
           name: v1.24.9---vmware.1-tkg.4
     nodePools:
     - name: worker-nodepool-a1
       replicas: 3
       storageClass: tkg-storage-policy
       vmClass: guaranteed-large
       tkr:
         reference:
           name: v1.24.9---vmware.1-tkg.4
```

```
      volumes:
      - name: containerd
        mountPath: /var/lib/containerd
        capacity:
          storage: 50Gi
      - name: kubelet
        mountPath: /var/lib/kubelet
        capacity:
          storage: 50Gi
    - name: worker-nodepool-a2
      ...
  settings:
    ...
```

# Supervisar el estado de los clústeres de TKG mediante vSphere Client

Puede supervisar el estado de los clústeres de TKG mediante vSphere Client.

Procedimiento

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 En el **Menú**, seleccione la vista **Hosts y clústeres**.

3 Expanda los objetos de **Centro de datos > Clúster** en los que se crea Supervisor.

4 Expanda el grupo de recursos **Espacios de nombres**.

5 Seleccione el espacio de nombres de vSphere en el que implementó el clúster de Tanzu Kubernetes.

   Cada clúster de TKG aparece como una carpeta dentro de su grupo de recursos de espacio de nombres. Cada instancia de TKG se representa gráficamente con el icono de tres hexágonos junto a su nombre.

6 Cambie a la vista **Menú > Máquinas virtuales y plantillas**.

   Dentro de la carpeta del clúster, verá las máquinas virtuales que conforman los nodos del clúster.

7 Seleccione el espacio de nombres de vSphere y, a continuación, seleccione la pestaña **Cálculo**.

8 En **Recursos de VMware**, seleccione **Tanzu Kubernetes**.

   Se mostrará cada clúster de TKG implementado en este espacio de nombres de vSphere.

# Supervisar el estado de los clústeres de TKG mediante kubectl

Puede supervisar el estado de los clústeres de TKG aprovisionados mediante kubectl.

### Procedimiento

1 Realice la autenticación con Supervisor.

2 Cambie al espacio de nombres de vSphere en el que se ejecuta el clúster.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 Vea una lista de los clústeres de Tanzu Kubernetes que se ejecutan en el espacio de nombres.

```
kubectl get tanzukubernetesclusters
```

Este comando devuelve el estado del clúster.

4 Vea los detalles del clúster.

```
kubectl describe tanzukubernetescluster <cluster-name>
```

El comando devuelve los detalles del clúster. En la sección Estado del resultado del comando, se muestra información detallada del clúster.

```
...
Status:
  Addons:
    Cni:
      Name:    calico
      Status:  applied
    Csi:
      Name:    pvcsi
      Status:  applied
    Psp:
      Name:    defaultpsp
      Status:  applied
  Cloudprovider:
    Name:  vmware-guest-cluster
  Cluster API Status:
    API Endpoints:
      Host:  10.161.90.22
      Port:  6443
    Phase:   provisioned
  Node Status:
    test-tanzu-cluster-control-plane-0:                  ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:       ready
  Phase:                                                 running
  Vm Status:
    test-tanzu-cluster-control-plane-0:                  ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:       ready
Events:                                                  <none>
```

5 Ejecute otros comandos `kubectl` para ver más detalles del clúster. Consulte Comprobar las operaciones del clúster de TKG mediante Kubectl.

# Comprobar la preparación del clúster de TKG mediante Kubectl

Cuando el controlador de TKG aprovisiona un clúster de TKG, se informan varias condiciones de estado que pueden servir para obtener información directa sobre los aspectos clave del estado de la máquina.

## Comprobar la preparación del clúster de TKG

Puede utilizar las condiciones de preparación del clúster de TKG para determinar qué fase o componente no está listo, si es que hay alguno.

Cuando haya comprobado la preparación del clúster, y para diagnosticar más detalles, puede utilizar vSphereCluster y las condiciones de máquina para ver más detalles del error.

Para comprobar la preparación de un clúster de TKG:

1 Inicie sesión en Supervisor.

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de destino. Por ejemplo:

```
kubectl config use-context tkg-cluster-ns
```

3 Ejecute el comando `kubectl get tkc -o yaml` o `kubectl get cluster -o yaml` según el tipo de clúster de TKG.

   El comando devuelve la preparación de los componentes del clúster. Consulte las secciones siguientes para obtener descripciones de los distintos estados de preparación.

## Condición y motivos de ControlPlaneReady

La tabla enumera y describe la condición `ControlPlaneReady`.

### Tabla 8-2. Condición ControlPlaneReady

| Tipo de condición | Descripción |
| --- | --- |
| `ControlPlaneReady` | Informa sobre si los nodos del plano de control están listos y en funcionamiento para el clúster. |

La tabla enumera y describe los motivos por los que la condición `ControlPlaneReady` puede ser falsa.

### Tabla 8-3. Motivos falsos de ControlPlaneReady

| Motivo | Descripción |
| --- | --- |
| `WaitingForClusterInfrastructure` | Indica que el clúster está esperando los requisitos previos necesarios para ejecutar máquinas como un equilibrador de carga. Este motivo solo se utiliza si InfrastructureCluster no informa de su propia condición de preparación. |
| `WaitingForControlPlaneInitialized` | Indica que se está inicializando el primer nodo de plano de control. |
| `WaitingForControlPlane` | Refleja la condición de KubeadmControlPlane. Este motivo se utiliza si KubeadmControlPlane no informa de su propia condición de preparación. |
| Esperando a que la infraestructura del clúster esté preparada | Mensaje que indica que el clúster está esperando los requisitos previos necesarios para ejecutar máquinas, como redes y equilibradores de carga. |

## Condición y motivos de NodesHealthy

La tabla enumera y describe la condición `NodesHealthy`.

### Tabla 8-4. Condición de NodesHealthy

| Tipo de condición | Descripción |
| --- | --- |
| `NodesHealthy` | Informa del estado de los nodos de TanzuKubernetesCluster. |

La tabla enumera y describe el motivo por el que la condición `NodesHealthy` no es verdadera.

### Tabla 8-5. Motivo falso de NodesHealthy

| Motivo | Descripción |
| --- | --- |
| `WaitingForNodesHealthy` | Documenta que no todos los nodos están en buen estado. |

## Condiciones y motivos de los complementos

En la tabla se enumeran y se describen las condiciones relacionadas con los componentes de complemento del clúster.

### Tabla 8-6. Condiciones de los complementos

| Tipo de condición | Descripción |
| --- | --- |
| `AddonsReady` | Resumen de condiciones de los complementos de TanzuKubernetesCluster (CoreDNS, KubeProxy, CSP, CPI, CNI, AuthSvc) . |
| `CNIProvisioned` | Documenta el estado del complemento de la interfaz de red de contenedor (CNI) de TanzuKubernetesCluster . |

### Tabla 8-6. Condiciones de los complementos (continuación)

| Tipo de condición | Descripción |
| --- | --- |
| `CSIProvisioned` | Documenta el estado del complemento de la interfaz de almacenamiento de contenedor (CSI) de TanzuKubernetesCluster. |
| `CPIProvisioned` | Documenta el estado de la interfaz de proveedor de nube (CPI) de TanzuKubernetesCluster. |
| `KubeProxyProvisioned` | Documenta el estado del complemento KubeProxy de TanzuKubernetesCluster. |
| `CoreDNSProvisioned` | Documenta el estado del complemento CoreDNS de TanzuKubernetesCluster. |
| `AuthServiceProvisioned` | Documenta el estado del complemento AuthService de TanzuKubernetesCluster. |
| `PSPProvisioned` | Documenta el estado de PodSecurityPolicy. |

La tabla enumera y describe los motivos por los que las condiciones del complemento no son verdaderas. Para solucionar problemas en los síntomas que provocan las advertencias, consulte Capítulo 21 Solucionar problemas de clústeres de servicio TKG.

### Tabla 8-7. Motivos falsos de complementos

| Motivo | Gravedad | Descripción |
| --- | --- | --- |
| `AddonsReconciliationFailed` | N/C | Motivo resumido de todos los errores de reconciliación de los complementos. |
| `CNIProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento CNI de los documentos. |
| `CSIProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento CSI de los documentos. |
| `CPIProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento CPI de los documentos. |
| `KubeProxyProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento KubeProxy de los documentos. |
| `CoreDNSProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento de CoreDNS de los documentos. |
| `AuthServiceProvisioningFailed` | Advertencia | No se pudo crear ni actualizar el complemento de AuthService de los documentos. |

### Tabla 8-7. Motivos falsos de complementos (continuación)

| Motivo | Gravedad | Descripción |
| --- | --- | --- |
| `AuthServiceUnManaged` | | El controlador no administra AuthService de los documentos. |
| `PSPProvisioningFailed` | Advertencia | No se pudieron crear ni actualizar los complementos de PodSecurityPolicy de los documentos. |

## Otras condiciones y motivos

En la tabla, se enumeran y describen las condiciones para la sincronización de StorageClass y RoleBinding, la reconciliación de recursos de ProviderServiceAccount, ServiceDiscovery y la compatibilidad con clústeres de TKG 2.0.

### Tabla 8-8. Otras condiciones

| Condición | Descripción |
| --- | --- |
| `StorageClassSynced` | Documenta el estado de sincronización de StorageClass del clúster supervisor al clúster de carga de trabajo. |
| `RoleBindingSynced` | Documenta el estado de sincronización de RoleBinding del clúster supervisor al clúster de carga de trabajo. |
| `ProviderServiceAccountsReady` | Documenta el estado de las cuentas de servicio del proveedor y se crean los Roles, RoleBindings y Secrets relacionados. |
| `ServiceDiscoveryReady` | Documenta el estado de los descubrimientos del servicio. |
| `TanzuKubernetesReleaseCompatible` | Indica si TanzuKubernetesCluster es compatible con TanzuKubernetesRelease. |

La tabla enumera y describe los motivos por los que otras condiciones no son verdaderas.

### Tabla 8-9. Otros motivos

| Motivo | Descripción |
| --- | --- |
| `StorageClassSyncFailed` | Informa que la sincronización de StorageClass ha fallado. |
| `RoleBindingSyncFailed` | Informa que la sincronización de RoleBinding ha fallado. |
| `ProviderServiceAccountsReconciliationFailed` | Informa que la reconciliación de recursos relacionados con las cuentas de servicio del proveedor ha fallado. |
| `SupervisorHeadlessServiceSetupFailed` | Documenta que la configuración del servicio sin cabecera para el servidor API del clúster supervisor ha fallado. |

# Comprobar el estado de la máquina del clúster de TKG con Kubectl

Cuando el controlador de TKG aprovisiona un clúster de carga de trabajo en Supervisor, se informan varias condiciones de estado que pueden servir para obtener información directa sobre los aspectos clave del estado del clúster.

## Acerca de las condiciones de estado de las máquinas

Un clúster de TKG incluye varias partes móviles, todas controladas por controladoras independientes pero relacionadas que funcionan de forma conjunta para compilar y mantener un conjunto de nodos de Kubernetes. Los objetos `TanzuKubernetesCluster` and `Cluster` proporcionan condiciones de estado que aportan información detallada sobre el estado de la máquina.

## Comprobar el estado de la máquina

Para comprobar el estado de una máquina de clúster de TKG:

1. Conéctese al Supervisor e inicie sesión.
2. Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

   ```
   kubectl config use-context CLUSTER-NAME
   ```

3. Ejecute el comando `kubectl describe machine`.

   El comando devuelve el estado de los nodos de máquina virtual que componen el clúster. Cuando una condición de la máquina como `InfrastructureReady` es `True` y `Ready`, el aspecto de la máquina será correcto. Sin embargo, si la condición de una máquina es `False`, quiere decir que la máquina no está lista. Consulte la lista de condiciones de estado de las máquinas para obtener descripciones de cada tipo de condición de máquina.

4. En ese caso, si la máquina no está lista, ejecute el siguiente comando y determine qué problema hay en la infraestructura:

   ```
   kubectl describe vspheremachine
   ```

## Lista de condiciones de mantenimiento de máquinas

En la tabla, se enumeran y definen las condiciones de estado de las máquinas disponibles para un clúster de TKG.

| Condición | Descripción |
| --- | --- |
| `ResourcePolicyReady` | Notifica la creación de una directiva de recursos. |
| `ResourcePolicyCreationFailed` | Se indica cuando se producen errores durante la creación de ResourcePolicy. |

| Condición | Descripción |
| --- | --- |
| ClusterNetworkReady | Notifica el aprovisionamiento correcto de una red de clústeres. |
| ClusterNetworkProvisionStarted | Se indica mientras se espera a que la red del clúster esté lista. |
| ClusterNetworkProvisionFailed | Se indica cuando se producen errores durante el aprovisionamiento de la red. |
| LoadBalancerReady | Informa sobre la reconciliación correcta de un Endpoint de plano de control estático. |
| LoadBalancerCreationFailed | Se indica cuando se produce un error al crear los recursos relacionados con el equilibrador de carga. |
| WaitingForLoadBalancerIP | Se indica mientras se espera a que haya una dirección IP del equilibrador de carga. |
| VMProvisioned | Informa de que se ha creado una máquina virtual, se ha encendido y se le ha asignado una dirección IP. |
| WaitingForBootstrapData | Se indica cuando una vSphereMachine espera a que el script de arranque esté listo antes de iniciar el proceso de aprovisionamiento. |
| VMCreationFailed | Informa de que ha habido un error en la creación del CRD de la máquina virtual o el correspondiente ConfigMap de arranque. |
| VMProvisionStarted | Se indica cuando una máquina virtual se encuentra en el proceso de creación. |
| PoweringOn | Se indica cuando una máquina virtual ejecuta en ese momento la secuencia de encendido. |
| WaitingForNetworkAddress | Se indica cuando se espera a que se active la configuración de red de la máquina. |
| WaitingForBIOSUUID | Se indica cuando se espera a que la máquina tenga un UUID de BIOS. |

## Campos de condición

Cada condición puede contener varios campos.

| | |
| --- | --- |
| Type | Describe el tipo de condición. Por ejemplo, ResourcePolicyReady. En el caso de la condición Ready, se trata de un resumen de todas las demás condiciones. |
| Status | Describe el estado del tipo.<br>Los estados pueden ser True, False o Unknown. |
| Severity | Clasificación de Reason.<br>Info significa que se está realizando la reconciliación.<br>Warning significa que es posible que haya algo mal y vuelva a intentarlo.<br>Error significa que se ha producido un error y hay que llevar a cabo una acción manual para resolverlo. |

| | |
|---|---|
| `Reason` | Proporciona un motivo por el cual el estado es `False`. Puede ser que haya que esperar a que esté listo o a que se indique el motivo de un error. Por lo general, se produce cuando el estado es `False`. |
| `Message` | Información de lenguaje natural que explica el significado de `Reason`. |

# Comprobar el estado del clúster de TKG con Kubectl

Cuando el controlador de TKG aprovisiona un clúster de carga de trabajo, se informan varias condiciones de estado que pueden servir para obtener información directa sobre los aspectos clave del estado del clúster.

## Acerca de las condiciones de estado del clúster

Un clúster de TKG aprovisionado incluye varias partes móviles, todas controladas por controladoras independientes pero relacionadas que funcionan de forma conjunta para compilar y mantener un conjunto de nodos de Kubernetes. Los objetos `TanzuKubernetesCluster` y `Cluster` proporcionan condiciones de estado que aportan información detallada sobre el estado de la máquina y del clúster.

## Comprobar estado del clúster

Para comprobar el estado de un clúster de TKG:

1 Ejecute el comando `kubectl describe cluster`.

Si el estado es listo, quiere decir que tanto la infraestructura del clúster como el plano de control del clúster están listos. Por ejemplo:

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     Ready
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     ControlPlaneReady
    Last Transition Time:     2020-11-24T21:31:34Z
    Status:                   True
    Type:                     InfrastructureReady
```

Sin embargo, si la condición de un clúster es "false", quiere decir que el clúster no está listo y un campo de mensaje describe lo que está mal. Por ejemplo, a continuación se muestra que el estado es "False" y el motivo por el que la infraestructura no está lista:

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   False
```

```
    Type:                    Ready
    Last Transition Time:    2020-11-24T21:37:32Z
    Status:                  True
    Type:                    ControlPlaneReady
    Last Transition Time:    2020-11-24T21:31:34Z
    Status:                  False
    Type:                    InfrastructureReady
```

2 Si el clúster no está listo, ejecute el siguiente comando para determinar qué problema hay en la infraestructura del clúster:

```
kubectl describe vspherecluster
```

## Lista de condiciones de estado del clúster

En la tabla se enumeran y definen las condiciones de estado disponibles para un clúster de TKG.

| Condición | Descripción |
| --- | --- |
| `Ready` | Resume el estado operativo de un objeto de API del clúster. |
| `Deleting` | El estado no es true porque el objeto subyacente se está eliminando en este momento. |
| `DeletionFailed` | El estado no es true debido a que el objeto subyacente detectó problemas durante la eliminación. Es una advertencia, porque el reconciliador volverá a intentar la eliminación. |
| `Deleted` | El estado no es true porque el objeto subyacente se eliminó. |
| `InfrastructureReady` | Informa de un resumen del estado actual del objeto de infraestructura definido para este clúster. |
| `WaitingForInfrastructure` | Se indica cuando un clúster espera a que la infraestructura subyacente esté disponible. NOTA: Esta condición se utiliza como reserva cuando la infraestructura no notifica que esté lista. |
| `ControlPlaneReady` | Se indica cuando el plano de control del clúster está listo. |
| `WaitingForControlPlane` | Se indica cuando un clúster espera a que el plano de control esté disponible. NOTA: Esta condición se utiliza como reserva cuando el plano de control no notifica que esté listo. |

## Campos de condición

Cada condición puede contener varios campos.

| | |
| --- | --- |
| `Type` | Describe el tipo de condición. Por ejemplo, `ControlPlaneReady`. En el caso de la condición `Ready`, se trata de un resumen de todas las demás condiciones. |
| `Status` | Describe el estado del tipo.<br>Los estados pueden ser `True`, `False` o `Unknown`. |

| | |
|---|---|
| `Severity` | Clasificación de `Reason`.<br>`Info` significa que se está realizando la reconciliación.<br>`Warning` significa que es posible que haya algo mal y vuelva a intentarlo.<br>`Error` significa que se ha producido un error y hay que llevar a cabo una acción manual para resolverlo. |
| `Reason` | Proporciona un motivo por el cual el estado es `False`. Puede ser que haya que esperar a que esté listo o a que se indique el motivo de un error. Por lo general, se produce cuando el estado es `False`. |
| `Message` | Información de lenguaje natural que explica el significado de `Reason`. |

# Comprobar el estado del volumen del clúster de TKG con Kubectl

Es posible comprobar el estado de mantenimiento de volumen persistente en un estado enlazado de un clúster de TKG.

Para cada volumen persistente en un estado enlazado, el estado de mantenimiento aparece en el campo `Annotations: volumehealth.storage.kubernetes.io/messages:` de la notificación de volumen persistente enlazada al volumen persistente. Existen dos valores posibles para el estado de mantenimiento.

| Estado de mantenimiento | Descripción |
|---|---|
| Accesible | Puede accederse al volumen persistente y está disponible para su uso. |
| Inaccesible | No puede accederse al volumen persistente y no puede usarse. El volumen persistente se vuelve inaccesible si los hosts que se conectan al almacén de datos no pueden acceder al almacén de datos que almacena el volumen. |

### Procedimiento

1 Inicie sesión en el clúster de TKG mediante Kubectl.

2 Cree una notificación de volumen persistente (Persistent Volume Claim, PVC).

a Cree un archivo YAML que contenga la configuración de notificación de volumen persistente.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 2Gi
```

b Aplique la notificación de volumen persistente al clúster de Kubernetes.

```
kubectl apply -f pvc_name.yaml
```

Este comando crea un volumen persistente de Kubernetes y un volumen de vSphere con un disco virtual de respaldo que cumple con los requisitos de almacenamiento de la notificación.

c Compruebe si la notificación de volumen persistente está enlazada a un volumen.

```
kubectl get pvc my-pvc
```

El resultado muestra que la notificación de volumen persistente y el volumen se encuentran enlazados.

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

3 Compruebe el estado de mantenimiento del volumen.

Ejecute el siguiente comando para comprobar la anotación del estado del volumen de la notificación de volumen persistente enlazada al volumen persistente.

```
kubectl describe pvc my-pvc
```

En los siguientes resultados de ejemplo, el campo `volumehealth.storage.kubernetes.io/messages` muestra el estado de mantenimiento como accesible.

```
Name:          my-pvc
Namespace:     test-ns
StorageClass:  gold
Status:        Bound
Volume:        my-pvc
```

```
Labels:         <none>
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: csi.vsphere.vmware.com
                volumehealth.storage.kubernetes.io/messages: accessible
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       2Gi
Access Modes:   RWO
VolumeMode:     Filesystem
```

# Supervisar el estado del volumen en un clúster de Tanzu Kubernetes Grid

Es posible comprobar el estado de mantenimiento de un volumen persistente en un estado enlazado.

Para cada volumen persistente en un estado enlazado, el estado de mantenimiento aparece en el campo `Annotations: volumehealth.storage.kubernetes.io/messages:` de la notificación de volumen persistente enlazada al volumen persistente. Existen dos valores posibles para el estado de mantenimiento.

| Estado de mantenimiento | Descripción |
| --- | --- |
| Accesible | Puede accederse al volumen persistente y está disponible para su uso. |
| Inaccesible | No puede accederse al volumen persistente y no puede usarse. El volumen persistente se vuelve inaccesible si los hosts que se conectan al almacén de datos no pueden acceder al almacén de datos que almacena el volumen. |

### Procedimiento

1 Acceda al espacio de nombres en el entorno de Kubernetes de vSphere.

**2** Cree una notificación de volumen persistente (Persistent Volume Claim, PVC).

a Cree un archivo YAML que contenga la configuración de notificación de volumen persistente.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 2Gi
```

b Aplique la notificación de volumen persistente al clúster de Kubernetes.

```
kubectl apply -f pvc_name.yaml
```

Este comando crea un volumen persistente de Kubernetes y un volumen de vSphere con un disco virtual de respaldo que cumple con los requisitos de almacenamiento de la notificación.

c Compruebe si la notificación de volumen persistente está enlazada a un volumen.

```
kubectl get pvc my-pvc
```

El resultado muestra que la notificación de volumen persistente y el volumen se encuentran enlazados.

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

**3** Compruebe el estado de mantenimiento del volumen.

Ejecute el siguiente comando para comprobar la anotación del estado del volumen de la notificación de volumen persistente enlazada al volumen persistente.

```
kubectl describe pvc my-pvc
```

En los siguientes resultados de ejemplo, el campo `volumehealth.storage.kubernetes.io/messages` muestra el estado de mantenimiento como accesible.

```
Name:          my-pvc
Namespace:     test-ns
StorageClass:  gold
Status:        Bound
Volume:        my-pvc
```

```
Labels:         <none>
Annotations:    pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: csi.vsphere.vmware.com
                volumehealth.storage.kubernetes.io/messages: accessible
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       2Gi
Access Modes:   RWO
VolumeMode:     Filesystem
```

# Supervisar volúmenes persistentes mediante vSphere Client

Cuando los ingenieros de desarrollo y operaciones implementan una aplicación con estado que contiene una notificación de volumen persistente, la vSphere IaaS control plane crea un objeto de volumen persistente y un disco virtual persistente coincidente. Como administrador de vSphere, puede revisar los detalles del volumen persistente en vSphere Client. También puede supervisar el estado de mantenimiento y el cumplimiento de almacenamiento.

### Procedimiento

1 En vSphere Client, desplácese hasta el espacio de nombres que tiene los volúmenes persistentes.

   a En el menú Inicio de vSphere Client, seleccione **Administración de cargas de trabajo**.

   b Haga clic en la pestaña **Espacios de nombres** y seleccione un espacio de nombres de la lista.

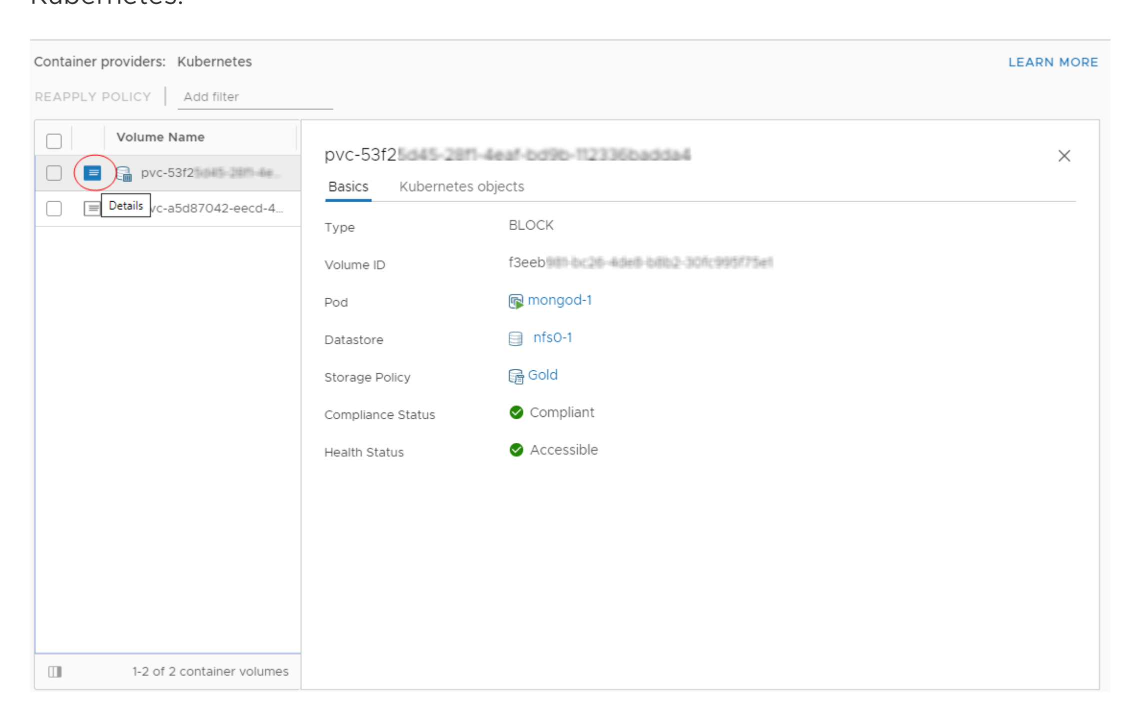
2 Haga clic en la pestaña **Almacenamiento** y, a continuación, en **Notificaciones de volumen persistente**.

   En vSphere Client, se enumeran todos los objetos de notificación de volumen persistente y los volúmenes correspondientes disponibles en el espacio de nombres.

3 Para ver los detalles de una notificación de volumen persistente seleccionada, haga clic en el nombre del volumen en la columna **Nombre de volumen persistente**.

4 En la página **Volúmenes contenedores**, compruebe el estado de mantenimiento del volumen y el cumplimiento de la directiva de almacenamiento.

   a Haga clic en el icono **Detalles** y alterne entre las pestañas **Conceptos básicos** y **Objetos de Kubernetes** para ver información adicional sobre el volumen persistente de Kubernetes.




   b Compruebe el estado de mantenimiento del volumen.

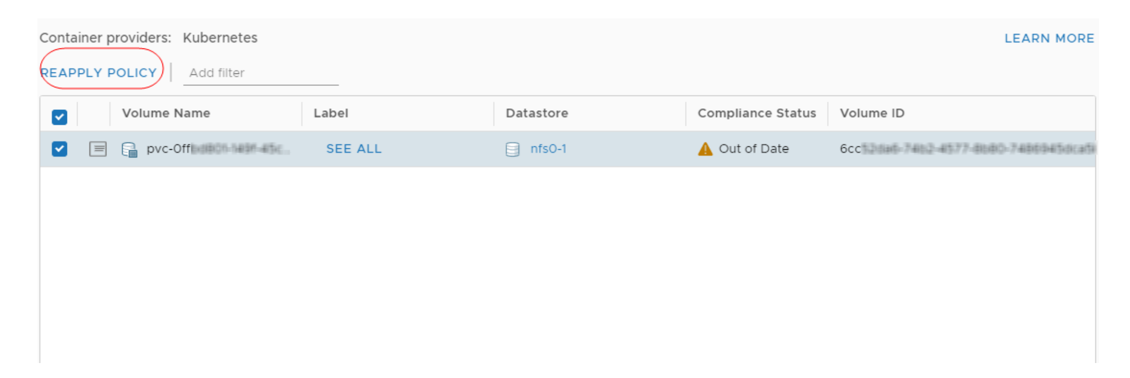
| Estado de mantenimiento | Descripción |
| --- | --- |
| Accesible | Puede accederse al volumen persistente y está disponible para su uso. |
| Inaccesible | No puede accederse al volumen persistente y no puede usarse. El volumen persistente se vuelve inaccesible si los hosts que se conectan al almacén de datos no pueden acceder al almacén de datos que almacena el volumen. |

c Compruebe el estado de cumplimiento del almacenamiento.

Puede ver una de las siguientes opciones en la columna **Estado de cumplimiento**.

| Estado de cumplimiento | Descripción |
|---|---|
| Conforme | El almacén de datos donde reside el disco virtual de respaldo del volumen tiene las capacidades de almacenamiento que requiere la directiva. |
| Desactualizado | Este estado indica que la directiva se editó, pero no se comunicaron los nuevos requisitos de almacenamiento al almacén de datos. Para comunicar los cambios, vuelva a aplicar la directiva en el volumen desactualizado. |
| No compatible | El almacén de datos cumple con los requisitos de almacenamiento especificados, pero actualmente no puede cumplir con la directiva de almacenamiento. Por ejemplo, el estado puede ser de no cumplimiento cuando los recursos físicos del almacén de datos no están disponibles. Puede lograr que el almacén de datos cumpla con los requisitos si realiza cambios en la configuración física del clúster de hosts, por ejemplo, si agrega hosts o discos al clúster. Si los recursos adicionales cumplen con la directiva de almacenamiento, el estado pasará a ser Cumplimiento. |
| No aplicable | La directiva de almacenamiento hace referencia a las capacidades del almacén de datos no admitidas por el almacén de datos. |

d Si el estado de cumplimiento es Desactualizado, seleccione el volumen y haga clic en **Volver a aplicar directiva**.




El estado pasará a ser Conforme.

# Obtener secretos de clúster de TKG mediante Kubectl

Los clústeres de TKG usan secretos para almacenar tokens, claves y contraseñas para operar.

## Lista de secretos de clústeres de TKG

Un secreto de Kubernetes es un objeto que almacena una pequeña cantidad de datos confidenciales, como una contraseña, un token o una clave SSH. Los administradores de clústeres de TKG pueden utilizar varios secretos al operar clústeres. En la tabla se enumeran y describen los secretos clave que podrían utilizar los administradores de clústeres.

**Nota** La lista no es exhaustiva; solo incluye esos secretos que habría que rotar manualmente o que habría que usar para acceder a nodos de clústeres con fines de solución de problemas.

| Secreto | Descripción |
| --- | --- |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-ccm-token-RANDOM` | Un token de cuenta de servicio utilizado por el administrador de controladoras de nube del proveedor de nube paravirtual para conectarse al espacio de nombres de vSphere. Para activar la rotación de esta credencial, elimine el secreto. |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-pvcsi-token-RANDOM` | Un token de cuenta de servicio utilizado por el complemento de CSI paravirtual para conectarse a espacio de nombres de vSphere. Para activar la rotación de esta credencial, elimine el secreto. |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-kubeconfig` | Un archivo kubeconfig que se puede utilizar para conectarse al plano de control del clúster como el usuario `kubernetes-admin`. Este secreto se puede utilizar para acceder a un clúster y solucionar los problemas que surjan en él cuando la autenticación de vCenter Single Sign-On no esté disponible. |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-ssh` | Una clave privada SSH que se puede utilizar para conectarse a cualquier nodo del clúster como `vmware-system-user`. Este secreto se puede utilizar para usar SSH en cualquier nodo del clúster y solucionar los problemas. |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-ssh-password` | Una contraseña que se puede utilizar para conectarse a cualquier nodo del clúster como `vmware-system-user`. |
| `*TANZU-KUBERNETES-CLUSTER-NAME*-ca` | El certificado de CA raíz para el plano de control del clúster de Tanzu Kubernetes que `kubectl` utiliza a fin de conectarse de forma segura al servidor de API de Kubernetes. |

# Comprobar las redes del clúster de TKG con Kubectl

El sistema aprovisiona clústeres de TKG con redes predeterminadas para nodos, pods y servicios. Puede comprobar las redes del clúster mediante los comandos kubectl personalizados.

## Comandos personalizados para comprobar las redes de clústeres de TKG

Consulte los siguientes comandos para comprobar las redes del clúster.

Estos comandos deben ejecutarse desde el espacio de nombres de vSphere, donde se aprovisiona el clúster de TKG. Por ejemplo:

```
kubectl config use-context tkg2-cluster-ns
```

## Tabla 8-10. Comandos kubectl personalizados para verificar las redes de clústeres

| Comando | Descripción |
| --- | --- |
| Comando<br>`kubectl get tkgserviceconfigurations`<br>Resultado de ejemplo<br>`NAME                          DEFAULT CNI`<br>`tkg-service-configuration   antrea` | Devuelve la CNI predeterminada, que es `antrea` a menos que se cambie.<br>La CNI predeterminada se utiliza para la creación del clúster, a menos que se anule explícitamente en el YAML del clúster. |
| Comando<br>`kubectl get virtualnetwork`<br>Resultado de ejemplo<br>`NAME                   SNAT`<br>`READY   AGE`<br>`tkgs-cluster-12-vnet   10.191.152.133`<br>`True    4h3m` | Devuelve la red virtual para los nodos del clúster.<br>Se usa para comprobar que la dirección IP de la traducción de direcciones de red (Network Address Translation, SNAT) de origen esté asignada. |
| Comando<br>`kubectl get virtualmachines -o wide`<br>Resultado de ejemplo<br>`NAME`<br>`POWERSTATE   CLASS`<br>`IMAGE`<br>`PRIMARY-IP    AGE`<br>`tkg2-cluster-12-control-plane-...`<br>`poweredOn    guaranteed-medium`<br>`ob-...-v1.23.8---vmware.1-tkg.1.b3d708a`<br>`10.244.0.66   4h6m`<br>`tkg2-cluster-12-worker-...`<br>`poweredOn    guaranteed-medium`<br>`ob-...-v1.22.9---vmware.1-tkg.1.b3d708a`<br>`10.244.0.68   4h3m`<br>`tkg2-cluster-12-worker-...`<br>`poweredOn    guaranteed-medium`<br>`ob-...-v1.21.6---vmware.1-tkg.1.b3d708a`<br>`10.244.0.67   4h3m` | Devuelve la interfaz de red virtual para los nodos del clúster.<br>Se usa para comprobar que la máquina virtual de cada nodo del clúster tiene una dirección IP asignada. |

Tabla 8-10. Comandos kubectl personalizados para verificar las redes de clústeres (continuación)

<table>
<tr><th>Comando</th><th>Descripción</th></tr>
<tr><td>Comando<br><code>kubectl get virtualmachineservices</code><br>Resultado de ejemplo<br><code>NAME<br>TYPE           AGE<br>tkg2-cluster-12-control-plane-service<br>LoadBalancer   3h53m</code></td><td>Devuelve el servicio de la máquina virtual para cada nodo del clúster.<br>Se usa para comprobar que el estado esté actualizado e incluya la dirección IP virtual (VIP) del equilibrador de carga.</td></tr>
<tr><td>Comando<br><code>kubectl get services -n NAMESPACE</code><br>Verificar mediante cURL<br><code>curl -k https://EXTERNAL-IP:PORT/healthz</code></td><td>Devuelve el equilibrador de carga del servicio de Kubernetes creado para acceder a la API del clúster.<br>Se usa para comprobar que se asignó una dirección IP externa.<br>Use <code>curl</code> para comprobar que se puede acceder a la API mediante la dirección IP externa y el puerto del servicio del equilibrador de carga.</td></tr>
<tr><td>Comando<br><code>kubectl get endpoints</code><br>Resultado de ejemplo<br><code>NAME<br>ENDPOINTS          AGE<br>tkg2-cluster-12-control-plane-service<br>10.244.0.66:6443   3h44m</code></td><td>Devuelve los nodos del plano de control (endpoints) del clúster. Se usa para comprobar que cada endpoint se cree e incluya en el grupo de endpoints.</td></tr>
</table>

# Comprobar las operaciones del clúster de TKG mediante Kubectl

Los clústeres de TKG se pueden administrar mediante comandos kubectl personalizados. Estos comandos están disponibles mediante recursos personalizados que crea el controlador de TKG.

## Comandos personalizados para administrar clústeres de TKG

En la tabla se enumeran y describen comandos kubectl para administrar clústeres de TKG.

Ejecute cada comando desde el contexto del espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG. La ejecución de estos comandos en el contexto del clúster no devuelve ninguna información.

### Tabla 8-11. Comandos personalizados para administrar clústeres de TKG

| Comando | Descripción |
| --- | --- |
| `kubectl get tanzukubernetescluster` | Enumera los TKC del espacio de nombres actual. |
| `kubectl get tkc` | Versión de formato corto del comando anterior. |
| `kubectl get cluster` | Devuelve los clústeres del espacio de nombres. |
| `kubectl describe tanzukubernetescluster CLUSTER-NAME` | Describe el clúster especificado mostrando el estado, la condición y los eventos expresados. Cuando se completa el aprovisionamiento, este comando muestra la IP virtual creada para el equilibrador de carga que presenta los endpoints de la API de Kubernetes. |
| `kubectl get cluster-api` | Enumera los recursos de la API del clúster que respaldan los clústeres en el espacio de nombres actual, incluidos los recursos del proyecto de la API del clúster y del proveedor de la API del clúster que Tanzu Kubernetes Grid Service utiliza. |
| `kubectl get tanzukubernetesreleases` | Lista de las versiones de Tanzu Kubernetes disponibles. |
| `kubectl get tkr` | Versión de formato corto del comando anterior. |
| `kubectl get tkr` *`v1.23.8---vmware.1-tkg.1.5417466`* `-o yaml` | Proporciona detalles sobre la versión de Tanzu Kubernetes con nombre. |
| `kubectl get virtualmachine` | Enumera los recursos de la máquina virtual que respaldan los nodos del clúster en el espacio de nombres actual. |
| `kubectl get vm` | Versión de formato corto del comando anterior. |
| `kubectl describe virtualmachine VIRTUAL-MACHINE-NAME` | Describe la máquina virtual especificada a través del estado actual y los eventos. |
| `kubectl describe virtualmachinesetresourcepolicy` | Enumera los recursos de directiva de recursos establecidos en la máquina virtual que respaldan el clúster en el espacio de nombres actual. Este recurso representa el grupo de recursos y la carpeta de objetos de vSphere que se usan para el clúster. |
| `kubectl get virtualmachineservice` | Enumera los recursos de servicio de la máquina virtual que respaldan los nodos del clúster en el espacio de nombres actual. Estos recursos son análogos a un servicio, pero para máquinas virtuales en lugar de pods. Los servicios de la máquina virtual se utilizan para proporcionar un equilibrador de carga para los nodos del plano de control de un clúster y para el proveedor de nube paravirtual con el fin de admitir un servicio de Kubernetes de tipo LoadBalancer dentro de un clúster. |

Tabla 8-11. Comandos personalizados para administrar clústeres de TKG (continuación)

| Comando | Descripción |
|---|---|
| `kubectl get vmservice` | Versión de formato corto del comando anterior. |
| `kubectl describe virtualmachineservice` *`VIRTUAL-MACHINE-SERVICE-NAME`* | Describe el servicio de la máquina virtual especificada a través del estado deseado del clúster, el estado actual y los eventos. |
| `kubectl get virtualmachineimage` | Enumera las imágenes de máquina virtual disponibles. |
| `kubectl get vmimage` | Versión de acceso directo del comando anterior. |
| `kubectl describe vmimage VM_IMAGE_NAME` | Vea los detalles de la imagen de la máquina virtual con nombre. |
| `kubectl get virtualnetwork` | Enumera los recursos de la red virtual en el espacio de nombres actual, incluidos los recursos que se utilizan para clústeres. Se crea una red virtual para cada espacio de nombres en el que se aprovisiona un clúster, así como para cada clúster en sí. |
| `kubectl get persistentvolumeclaim` | Enumera los recursos de notificación de volumen persistente en el espacio de nombres actual, incluidos los recursos utilizados para los clústeres de. |
| `kubectl get cnsnodevmattachment` | Enumera los recursos de asociación de máquinas virtuales del nodo de CNS en el espacio de nombres actual. Estos recursos representan la asociación de un volumen persistente administrado por CNS con una máquina virtual que actúa como el nodo de un clúster. |
| `kubectl get configmap` | Enumera los mapas de configuración en el espacio de nombres actual, incluidos los que se utilizan para crear nodos del clúster. Los mapas de configuración no están pensados para que el usuario pueda modificarlos, y cualquier cambio realizado se sobrescribirá. |
| `kubectl get secret` | Enumera los secretos del espacio de nombres actual, incluidos los que se utilizan para crear y administrar nodos del clúster. |

# Ver el estado del ciclo de vida del clúster de TKG

Puede ver el estado del ciclo de vida de los clústeres de TKG en el inventario de vSphere y con kubectl.

## Estado del ciclo de vida del clúster de TKG en vSphere

En la tabla se incluye y se describe la información del estado de los clústeres de TKG que aparece en el inventario de vSphere.

### Tabla 8-12. Estado del clúster de TKG en el inventario de vSphere

| Campo | Descripción | Ejemplo |
| --- | --- | --- |
| Nombre | Nombre del clúster que define el usuario. | `tkg2-cluster-01` |
| Hora de creación | Fecha y hora de creación del clúster. | `Mar 17, 2022, 11:42:46 PM` |
| Fase | El estado del ciclo de vida del clúster. | `creating` |
| Número de trabajos | Cantidad de nodos de trabajo del clúster. | `1` o `2` o `5` |
| Versión de distribución | Versión del software de Kubernetes que se ejecuta en el clúster. | `v1.22.6+vmware.1-tkg.1.7144628` |
| Dirección del plano de control | Dirección IP del equilibrador de carga del plano de control del clúster. | `192.168.123.2` |

## El estado del ciclo de vida del clúster de TKG en kubectl

En la tabla se incluye y se describe la información del estado del clúster de TKG que aparece en kubectl.

### Tabla 8-13. Estado del clúster de TKG en kubectl

| Campo | Descripción | Ejemplo |
| --- | --- | --- |
| NAME | Nombre del clúster. | `tkg2-cluster-01` |
| CONTROL PLANE | Cantidad de nodos del plano de control del clúster. | `3` |
| WORKER | Cantidad de nodos de trabajo del clúster. | `5` |
| DISTRIBUTION | Versión de Kubernetes que ejecuta el clúster. | `v1.22.6+vmware.1-tkg.1.5b5608b` |
| AGE | Cantidad de días de ejecución del clúster. | `13d` |
| PHASE | El estado del ciclo de vida del clúster. | `running` |

## Estado de la fase del ciclo de vida de los clústeres

En la tabla se enumera y se describe el estado de cada fase del ciclo de vida de un clúster.

### Tabla 8-14. Estado de la fase del ciclo de vida de los clústeres

| Fase | Descripción |
| --- | --- |
| `creating` | El aprovisionamiento de clústeres puede comenzar, el plano de control se está creando o el plano de control se crea pero no se inicializa. |
| `deleting` | El clúster se está eliminando. |
| `failed` | Se produjo un error en la creación del plano de control del clúster y es probable que el usuario tenga que intervenir de alguna forma. |

Tabla 8-14. Estado de la fase del ciclo de vida de los clústeres (continuación)

| Fase | Descripción |
| --- | --- |
| `running` | La infraestructura se crea y se configura, y el plano de control se inicializa por completo. |
| `updating` | El clúster se está actualizando. |

# Ver la jerarquía de recursos de un clúster de TKG mediante Kubectl

Puede ver la jerarquía de recursos de un clúster de TKG mediante kubectl. Al ver la lista completa de recursos del clúster, podrá determinar los recursos que pueden estar causando problemas.

#### Procedimiento

1 Conéctese al Supervisor e inicie sesión.

2 Cambie el contexto al espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG.

```
kubectl config use-context tkg2-cluster-ns
```

3 Ejecute el siguiente comando para ver el recurso de clúster de la API del clúster.

```
kubectl describe clusters.cluster.x-k8s.io CLUSTER-NAME
```

Este comando devuelve la jerarquía de recursos del clúster designado de la siguiente forma: espacio de nombres, versión de la API, versión del recurso.

# Configurar MachineHealthCheck para clústeres v1beta1

En este tema se describe cómo configurar MachineHealthCheck para clústeres de Servicio TKG aprovisionados mediante la API v1beta1.

## MachineHealthCheck para clústeres v1beta1

MachineHealthCheck es un recurso de la API de clúster de Kubernetes que define las condiciones para corregir máquinas en mal estado. En Kubernetes, una máquina es un recurso personalizado que puede ejecutar kubelet. En vSphere IaaS control plane, un recurso de máquina de Kubernetes está respaldado por una máquina virtual de vSphere. Para obtener más información, consulte la documentación ascendente.

Al aprovisionar un clúster mediante Servicio TKG, el sistema crea objetos predeterminados MachineHealthCheck, uno para todos los planos de control y otro para cada implementación de máquina. A partir de vSphere 8 Update 3, se pueden configurar las comprobaciones de estado de las máquinas para los clústeres v1beta1. La configuración admitida incluye lo siguiente:

- maxUnhealthy

- nodeStartupTimeout
- unhealthyConditions
- unhealthyRange

En la tabla se describen las operaciones admitidas de comprobación de estado de máquinas.

**Tabla 8-15. Comprobaciones de estado de máquinas**

| Campo | Valor | Descripción |
|---|---|---|
| `maxUnhealthy` | string<br>Número absoluto o porcentaje | No se realizará la corrección cuando el número de máquinas en mal estado supere el valor. |
| `nodeStartupTimeout` | string<br>Duración con el formato `XhXmXs` (horas, minutos, segundos) | Cualquier máquina que se esté creando que tarde más que el tiempo en unirse al clúster se considera con errores y se corregirá. |
| `unhealthyConditions` | matriz [] de tipos unhealthyConditions<br>Tipos de condición disponibles:<br>[`Ready`, `MemoryPressure`,`DiskPressure`, `PIDPressure`, `NetworkUnavailable`]<br>Estados de condición disponibles:<br>[`True`, `False`, `Unknown`] | Lista de condiciones que determinan si un nodo del plano de control se considera en mal estado. |
| `unhealthyRange` | string | Cualquier otra corrección solo se permite si el número de máquinas seleccionadas por el "selector" como en mal estado está dentro del rango de `unhealthyRange`.<br>Tiene prioridad sobre `maxUnhealthy`. Por ejemplo: "[3-5]" significa que la corrección solo se permitirá cuando (a) haya al menos 3 máquinas en mal estado (y) (b) haya como máximo 5 máquinas en mal estado. |

**Nota** Los objetos MachineHealthCheck se implementan para clústeres v1alpha3, pero no se pueden configurar. Para obtener más información, consulte Comprobar el estado de la máquina del clúster de TKG con Kubectl.

## Ejemplo de MachineHealthCheck

En el siguiente ejemplo, se configura `machineHealthCheck` para un `machineDeployment` determinado.

```
...
  topology:
    class: tanzukubernetescluster
    version: v1.28.8---vmware.1-fips.1-tkg.2
    controlPlane:
```

```
      machineHealthCheck:
        enable: true
        maxUnhealthy: 100%
        nodeStartupTimeout: 4h0m0s
        unhealthyConditions:
        - status: Unknown
          timeout: 5m0s
          type: Ready
       - status: "False"
          timeout: 12m0s
          type: Ready
      ...
    workers:
      machineDeployments:
      - class: node-pool
        failureDomain: np1
        machineHealthCheck:
          enable: true
          maxUnhealthy: 100%
          nodeStartupTimeout: 4h0m0s
          unhealthyConditions:
          - status: Unknown
            timeout: 5m0s
            type: Ready
          - status: "False"
            timeout: 12m0s
            type: Ready
```

## Revisar MachineHealthCheck mediante Kubectl

Para actualizar la `MachineHealthCheck` de un clúster v1beta1 después de que se haya aprovisionado, utilice el método `patch`.

**Precaución** Estas instrucciones proporcionan una orientación general sobre cómo aplicar revisiones a un clúster existente. Los valores que utilice dependerán del entorno y del clúster implementado en el que se apliquen revisiones. Considere la posibilidad de usar la CLI de Tanzu para aplicar revisiones a `MachineHealthCheck` de un clúster existente.

1 Obtenga `machineDeployment` a partir de la definición de recursos del clúster.

```
kubectl get cluster CLUSTER_NAME -o yaml
```

En la sección `spec.topology.workers.machineDeployments`, debería ver el valor que identifica cada `machineDeployment`.

2 Elimine el nodo de trabajo MachineHealthCheck.

```
kubectl patch cluster <Cluster Name> -n <cluster namespace>  --type json -p='{"op":
"replace", "path": "/spec/topology/workers/machineDeployments/<index>/machineHealthCheck",
"value":{"enable":false}}'
```

3 Elimine el plano de control MachineHealthCheck.

```
kubectl patch cluster <cluster-name> -n <cluster-namespace> --type json
-p='{"op": "replace", "path": "/spec/topology/controlPlane/machineHealthCheck", "value":
{"enable":false}}'
```

4 Cree o actualice el plano de control MachineHealthCheck con la configuración deseada.

```
kubectl patch cluster <cluster-name> -n <cluster-namespace> --type json
-p='[{"op": "replace", "path": "/spec/topology/controlPlane/machineHealthCheck",
"value":{"enable":true,"nodeStartupTimeout":"1h58m","unhealthyConditions":
[{"status":"Unknown","timeout":"5m10s","type":"Unknown"},
{"status":"Unknown","timeout":"5m0s","type":"Ready"}],"maxUnhealthy":"100%"}}]'
```

5 Cree o actualice el nodo de trabajo MachineHealthCheck con la configuración deseada.

```
kubectl patch cluster <cluster-name> -n <cluster-namespace> --type json -p='[{"op":
"replace", "path": "/spec/topology/workers/machineDeployments/<index>/machineHealthCheck",
"value":{"enable":true,"nodeStartupTimeout":"1h58m","unhealthyConditions":
[{"status":"Unknown","timeout":"5m10s","type":"Unknown"},
{"status":"Unknown","timeout":"5m0s","type":"Ready"}],"maxUnhealthy":"100%"}}]'
```

## Configurar MachineHealthCheck mediante la CLI de Tanzu

Puede utilizar la CLI de Tanzu a fin de configurar MachineHealthCheck para un clúster v1beta1.

Por ejemplo, ejecute el siguiente comando para crear o actualizar la configuración de MachineHealthCheck del plano de control.

```
tanzu cluster mhc control-plane set <cluster-name> --node-startup-timeout 2h7m10s
```

Ejecute el siguiente comando para comprobar si la configuración se actualizó y no se concilió.

```
tanzu cluster mhc control-plane get <cluster-name>
```

Ejecute el siguiente comando para crear o actualizar la configuración de MachineHealthCheck de la implementación de la máquina.

```
tanzu cluster mhc node set <cluster-name> --machine-deployment node-pool-1 --node-startup-
timeout 1h59m0s
```

Ejecute el siguiente comando para comprobar si la configuración se actualizó y no se concilió.

```
tanzu cluster mhc node get <cluster-name> -m <cluster-name>-node-pool-1-nr7r5
```

Además de obtener y establecer, el sistema admite la operación de eliminación. Por ejemplo:

Para el plano de control puede usar el siguiente comando:

```
tanzu cluster mhc control-plane delete <cluster-name>
```

Por el nodo puede usar el siguiente comando:

```
tanzu cluster mhc <cluster-name> --machine-deployment <machine deployment name>
```

# Actualizar clústeres de servicio TKG

9

En esta sección, se proporcionan instrucciones para actualizar los clústeres de servicio TKG.

Lea los siguientes temas a continuación:



## Información sobre el modelo de actualización gradual para clústeres de Servicio TKG

Los clústeres de Servicio TKG admiten un modelo de actualización gradual. Para iniciar una actualización gradual, puede cambiar la especificación del clúster. Algunas operaciones del sistema pueden iniciar una actualización gradual. Antes de actualizar el entorno, debe familiarizarse con el proceso de actualización gradual.

### Modelo de actualización gradual para clústeres TKGS después de Servicio TKG 3.0

A partir de Servicio TKG 3.0, la controladora TKG es independiente de vCenter Server y Supervisor. Consulte Usar Servicio TKG. La actualización de estos componentes no iniciará una actualización gradual de los clústeres de TKGS.

La actualización de la versión de Servicio TKG podría activar una actualización gradual de los clústeres de TKGS.

## Modelo de actualización gradual para clústeres de TKGS anteriores a Servicio TKG 3.0

La controladora TKG se ejecuta en el Supervisor. Cuando se actualiza el Supervisor, la controladora TKG se actualiza automáticamente si hay alguna actualización disponible. Cada actualización de la controladora TKG puede incluir actualizaciones para servicios de respaldo, como CNI, CSI y CPI, así como actualizaciones de configuración para los clústeres. Para admitir la compatibilidad, el sistema realiza comprobaciones previas y aplica la conformidad.

vSphere IaaS control plane utiliza un modelo de actualización gradual para los clústeres de TKG en el Supervisor. El modelo de actualización gradual garantiza un tiempo de inactividad mínimo durante el proceso de actualización. Las actualizaciones graduales incluyen la actualización de las versiones de software de Kubernetes, así como de la infraestructura y los servicios que respaldan los clústeres, como los recursos y las configuraciones de máquinas virtuales, los servicios y los espacios de nombres, y los recursos personalizados. Para que la actualización se realice correctamente, la configuración debe cumplir con varios requisitos de compatibilidad, de modo que el sistema aplique condiciones de reverificación a fin de garantizar que los clústeres estén listos para las actualizaciones y que dicho sistema sea compatible con la reversión si el clúster no se actualiza correctamente.

Para iniciar una actualización gradual de un clúster de TKG, cambie ciertos aspectos del manifiesto del clúster. El sistema también puede iniciar una actualización gradual del clúster. Por ejemplo, cuando se realiza una actualización de espacios de nombres de vSphere, el sistema propaga de inmediato las configuraciones actualizadas a todos los clústeres de carga de trabajo. Estas actualizaciones pueden activar una actualización gradual de los nodos del clúster. Un cambio en cualquiera de los elementos de configuración también puede iniciar una actualización gradual. Por ejemplo, si se cambia el nombre o se reemplaza el objeto `VirtualMachineImage` que corresponde a una versión de distribución, se inicia una actualización gradual, ya que el sistema intenta obtener todos los nodos que se ejecutan en la nueva imagen. Además, la actualización de Supervisor podrá activar una actualización gradual de los clústeres de carga de trabajo implementados allí. Por ejemplo, si se actualiza `vmware-system-tkg-controller-manager`, el sistema introduce nuevos valores en el generador de manifiestos, y la controladora inicia una actualización gradual para implementar esos valores.

El proceso de actualización gradual para reemplazar los nodos del clúster es similar a la actualización gradual de pods de una implementación de Kubernetes. Existen dos controladoras distintas que se encargan de realizar una actualización gradual de los clústeres de carga de trabajo: la controladora de complementos y la controladora de clúster. Dentro de esas dos controladoras hay tres etapas clave de una actualización gradual: la actualización de los complementos, la actualización del plano de control y la actualización de los nodos de trabajo. Estas etapas se producen en orden, con comprobaciones previas que impiden que un paso comience hasta que el paso anterior haya avanzado lo suficiente. Es posible que estos pasos se omitan si se determina que son innecesarios. Por ejemplo, una actualización podría solo afectar a los nodos de trabajo y, por tanto, no necesitaría actualizaciones de los planos de control ni de los complementos.

Durante el proceso de actualización, el sistema agrega un nuevo nodo de clúster y espera a que el nodo se conecte con la versión de Kubernetes de destino. A continuación, el sistema marca el nodo antiguo para su eliminación, pasa al siguiente nodo y repite el proceso. El nodo antiguo no se eliminará hasta que se eliminen todos los pods. Por ejemplo, si un pod se define con PodDisruptionBudgets que impide que un nodo se vacíe completamente, el nodo se acordona, pero no se elimina hasta que dichos pods se puedan expulsar. El sistema actualiza primero todos los nodos del plano de control y, a continuación, los nodos de trabajo. Durante una actualización, el estado del clúster cambia a "Actualizando". Una vez finalizado el proceso de actualización gradual, el estado del clúster cambia a "En ejecución".

Los pods en ejecución en un clúster no regulados por una controladora de replicación se eliminarán durante una actualización de la versión de Kubernetes como parte de la purga de nodos de trabajo que se realiza al actualizar el clúster. Esto sucede si una actualización de espacios de nombres de vSphere o de Supervisor activa la actualización del clúster de forma manual o automática. Los pods que no están regidos por una controladora de replicación incluyen aquellos pods que no se crean como parte de una especificación de ReplicaSet o de implementación. Para obtener más información, consulte Ciclo de vida del pod: duración del pod en la documentación de Kubernetes.

## Actualizaciones graduales iniciadas por el usuario

Es posible iniciar una actualización gradual de un clúster de TKG en el Supervisor. Para ello, se debe actualizar la versión de versión de Tanzu Kubernetes, actualizar la clase de máquina virtual y actualizar la clase de almacenamiento. Consulte uno de los siguientes temas para conocer más detalles.

- Actualizar un clúster de TKG mediante el cambio de la versión de TKR
- Actualizar un clúster de TKG mediante la edición de la clase de almacenamiento
- Actualizar un clúster de servicio TKG mediante la edición de la clase de Servicios
- Actualizar un clúster de TKG mediante la CLI de Tanzu

## Actualizaciones graduales iniciadas por el sistema

En cada versión del Supervisor, se pueden realizar cambios en uno o varios de los siguientes objetos:

- kubeadmcontrolplanetemplate/kubeadmcontrolplane
- kubeadmconfigtemplate/kubeadmconfig
- vspheremachinetemplate/vspheremachine (para vSphere 8.x)
- wcpmachinetemplate/wcpmachine (por vSphere 7.x)

Cuando se actualiza el Supervisor, los controladores principales de la API del clúster (CAPI) activan una actualización gradual en los clústeres de carga de trabajo de TKG para que coincidan con el estado deseado de los objetos anteriores en los clústeres de carga de trabajo en ejecución.

En vSphere IaaS control plane, la controladora de TKG que se ejecuta en el Supervisor genera y mantiene estos objetos sincronizados con el código del sistema. Esto significa que, cuando las controladoras se actualizan al código más reciente, los cambios que se producen en alguno de los objetos anteriores provocan una actualización gradual de los clústeres de TKG existentes. En resumen, los cambios en el código del sistema que afectan al Supervisor provocan actualizaciones graduales de los clústeres de TKG.

En la tabla se describen las condiciones en las que puede esperar una actualización gradual automatizada de los clústeres de carga de trabajo cada vez que el Supervisor se actualice.

| Escenario de actualización | Descripción |
| --- | --- |
| Actualizar desde cualquier versión de vCenter Server 7.x a cualquier versión de vCenter Server | Puede activar una actualización gradual de todos los clústeres de Tanzu Kubernetes.<br>La primera actualización de Supervisor después de una actualización de vCenter Server activa una actualización gradual. Por lo general, no se activa una actualización gradual al actualizar Supervisor en el mismo vCenter Server.<br>Compruebe las notas de la versión para obtener más información. |
| Actualizar desde cualquier versión de vCenter Server a cualquier versión de vCenter Server 8.x | Activará una actualización gradual de todos los clústeres de TKG porque se deben propagar los siguientes cambios de código:<br>■ Los proveedores de CAPI subyacentes deben moverse de CAPW a CAPV<br>■ Migrar los clústeres desde clústeres de CAPI sin clase a un clúster de CAPI con clase |
| Actualizar de la versión vCenter Server 8.0 GA (8.0.0) a las versiones vCenter Server 8.0.0b u 8.0.0c | Activará una actualización gradual de los clústeres de TKG especificados si se aplica alguno de los siguientes casos:<br>■ Cualquier clúster de TKG que utilizaba la configuración de proxy con una lista de noProxy que no está vacía.<br>■ Todos los clústeres de TKG si el servicio de registro de Harbor integrado se habilitó en Supervisor. |
| Actualización de la versión vSphere 8.0.0b a la versión vSphere 8.0.0c | No hay implementaciones automáticas de clústeres de carga de trabajo |
| Actualización de la versión vSphere 8.0.0c a la versión vSphere 8.0 Update 1 (8.0.1) | No hay implementaciones automáticas de clústeres de carga de trabajo |

| Escenario de actualización | Descripción |
|---|---|
| Actualizar desde cualquier versión de vSphere 8.x a la versión 8.0 U2 (8.0.2) | Esto provocará una actualización gradual en todos los TKC, ya que deben producirse los siguientes cambios:<br>■ vSphere 8.0 U2 contiene cambios de STIG a nivel de Kubernetes para las TKR de TKG 1.0 y TKG 2.0 en GCM como parte de la ClusterClass.<br>■ Dado que los TKC de la versión 1.23 y versiones posteriores son compatibles con 8.0U2, todos los clústeres se someterían a una actualización gradual. |
| Actualización de cualquier versión de vSphere 8.x anterior a la 8.0 U2 (8.0.2) a la versión 8.0 U2c | Esto provocará una actualización gradual en todos los TKC, ya que deben producirse los siguientes cambios:<br>■ 8.0 U2 contiene cambios de STIG a nivel de k8s para las TKR de TKG 1.0 y TKG 2.0 en GCM como parte de la ClusterClass.<br>■ Dado que los TKC de la versión 1.23 y versiones posteriores son compatibles con 8.0 P03, todos los clústeres se someterían a una actualización gradual. |

Además, al cambiar la biblioteca de contenido que aloja las imágenes de TKR se pueden activar actualizaciones graduales de los clústeres de TKG. Al añadir nuevas imágenes, ya sea a través de una suscripción o de forma manual, no se activa una actualización gradual de los clústeres de TKG. Sin embargo, si se cambia la biblioteca de contenido y se agregan imágenes con otros nombres, se activará una actualización gradual de todos los clústeres de TKG.

Por ejemplo, imagine un escenario en el que se utiliza una biblioteca de contenido suscrita que utiliza automáticamente los nombres de OVA definidos por el sistema. A continuación, se pasa a una biblioteca de contenido local y se rellena con los mismos archivos OVA, pero dándoles otros nombres. Esto activará una actualización gradual de todos los clústeres de TKG, ya que la biblioteca de contenido de reemplazo tiene los mismos archivos OVA, pero con otros nombres definidos por el usuario.

## Consideraciones sobre la actualización gradual para clústeres con varios grupos de nodos

Si utiliza clústeres de TKG con varios grupos de nodos, tenga en cuenta la siguiente información con respecto a las actualizaciones graduales.

**Grupos de nodos de trabajo**

Los grupos de nodos de trabajo se introdujeron con la API v1alpha2 de TKGS que se publicó con vSphere 7 U3. MachineDeployments de la API de clúster es el primitivo subyacente de Kubernetes de los grupos de nodos de trabajo.

ClusterClass se introdujo con la versión vSphere 8 de TKGS. Tanto la API v1alpha3 como v1beta1 se basan en ClusterClass. (v1alpha3 es una capa de abstracción encima de ClusterClass).

**Cómo se actualizan varios grupos de nodos durante una actualización gradual**

Cuando se actualiza un clúster de carga de trabajo de TKGS aprovisionado con varios grupos de nodos, el modelo de actualización gradual varía según la versión de vSphere que se utilice.

| vSphere | API de TKGS | Comportamiento de actualización |
| --- | --- | --- |
| TKGS de vSphere 7 | API v1alpha2 | Se actualizan varios grupos de nodos dentro del mismo clúster al mismo tiempo (simultáneamente) |
| TKGS de vSphere 8 | API v1alpha3 y API v1beta1 | Varios grupos de nodos dentro del mismo clúster se actualizan siguiendo un orden lógico (secuencialmente) |

### Consideraciones sobre prácticas recomendadas

Aprovisionar un clúster de TKGS de vSphere 8 con varios grupos de nodos idénticos no tiene ningún propósito en términos de dimensionamiento. Los grupos de nodos deben utilizarse para diferentes tamaños, clases de máquinas virtuales, versiones de TKr, etc. Evite utilizar varios grupos de nodos idénticos para burlar al sistema y actualizar los clústeres más rápido, ya que no funcionará.

Los Pod Disruption Budget son la forma adecuada de garantizar que las actualizaciones no interfieran con las aplicaciones en ejecución. La mejor forma de gestionar esto es establecer PodDisruptionBudgets en las cargas de trabajo (consulte https://kubernetes.io/docs/tasks/run-application/configure-pdb/). La API del clúster los respeta y no finalizará la máquina correspondiente si se superan los umbrales.

### Detalles de actualización gradual para clústeres TKGS de vSphere 8

Durante una actualización de la versión de un clúster TKGS de vSphere 8:

- Los nodos del plano de control se actualizan primero y, a continuación, los nodos de trabajo se consolidan en un nodo de trabajo a la vez empezando por el grupo de nodos de la zona A. Si se utilizan dos grupos de nodos, solo habrá 1 trabajo implementado a la vez.

Durante las actualizaciones de variables de configuración del clúster:

- Los nodos del plano de control se actualizan primero y, a continuación, se procede a actualizar un nodo de trabajo por grupo de nodos. Por ejemplo, si se utilizan dos grupos de nodos, habrá 2 nodos de trabajo actualizándose a la vez.

# Comprobar la compatibilidad del clúster TKGS para la actualización

Antes de actualizar un clúster de carga de trabajo de TKGS, debe comprobar su compatibilidad para la actualización. La compatibilidad debe comprobarse contra el Servicio TKG.

## Comprobar la compatibilidad con el Servicio TKG

Antes de actualizar un clúster de carga de trabajo, debe comprobar su compatibilidad para la actualización. Si un clúster no es compatible con el Servicio TKG, actualice la versión de Tanzu Kubernetes. Consulte las Notas de la versión para obtener más información sobre los TKr disponibles. Consulte también la matriz de interoperabilidad en línea.

Puede enumerar las versiones de Tanzu Kubernetes y ver su compatibilidad mediante el siguiente comando.

```
kubectl get tkr
```

La columna `COMPATIBLE` indica si esa versión de Tanzu Kubernetes es compatible con el Servicio TKG instalado. A partir de Servicio TKG versión 3.1, la columna TIPO también devuelve el estado de compatibilidad.

Si especifica el clúster TKGS, puede ver qué actualizaciones de TKr están disponibles.

Si utiliza la API v1alpha3:

```
kubectl get tkc <tkgs-cluster-name>
```

O bien, si utiliza la API v1beta1:

```
kubectl get cc <tkgs-cluster-name>
```

La columna `UPDATES AVAILABLE` indica si hay una actualización de Kubernetes disponible y la siguiente versiones de Tanzu Kubernetes recomendada que se utilizará. Por ejemplo:

```
kubectl get tkc tkg2-cluster-11-tkc
NAME                  CONTROL PLANE   WORKER   TKR NAME                         AGE
READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkg2-cluster-11-tkc   3               3        v1.25.7---vmware.3-fips.1-tkg.1   13d
True    True             [v1.26.5+vmware.2-fips.1-tkg.1]
```

Existen dos tipos de formatos de TKr: no heredados y heredados.

- Las TKr no heredadas están diseñados específicamente para vSphere 8.x y solo son compatibles con vSphere 8.x
- Las TKr heredadas utilizan un formato heredado que es compatible con vSphere 7.x y también con vSphere 8.x, pero solo para fines de actualización.

Para enumerar las TKr no heredadas:

```
kubectl get -l !run.tanzu.vmware.com/legacy-tkr
```

Para enumerar las TKr heredadas:

```
kubectl get -l !run.tanzu.vmware.com/legacy-tkr
```

# Actualizar un clúster de TKG mediante el cambio de la versión de TKR

Esta tarea se describe cómo actualizar la versión de versión de Tanzu Kubernetes para un clúster de TKG mediante la edición del manifiesto del clúster de TKG.

Para iniciar una actualización gradual de un clúster de TKGS, actualice la versión de la versión de Tanzu Kubernetes mediante el comando `kubectl edit`.

**Nota** No se puede utilizar el comando `kubectl apply` para actualizar la versión de TKR de un clúster implementado.

### Requisitos previos

Esta tarea requiere el uso del comando de edición kubectl. Este comando abre el manifiesto del clúster en el editor de texto definido por las variables de entorno KUBE_EDITOR o EDITOR. Al guardar el archivo, el clúster se actualiza con los cambios. Si desea configurar un editor para kubectl de modo que pueda ejecutar el comando `kubectl edit`, consulte Configurar un editor de texto para Kubectl.

### Procedimiento

1 Realice la autenticación con Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de carga de trabajo de destino.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 Obtenga la versión y el clúster de TKG de destino.

Clúster v1alpha3:

```
kubectl get tanzukubernetescluster
```

Clúster v1beta1:

```
kubectl get cluster
```

4 Enumere las versiones disponibles de Tanzu Kubernetes.

```
kubectl get tanzukubernetesreleases
```

5 Ejecute el siguiente comando para editar el manifiesto del clúster.

Clúster v1alpha3:

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

Clúster v1beta1:

```
kubectl edit cluster/CLUSTER-NAME
```

6 Edite el manifiesto mediante la actualización de la cadena de versión de Tanzu Kubernetes.

Por ejemplo, para un clúster v1alpaha3, cambie de TKR v1.25.7:

```
topology:
  controlPlane:
   replicas: 1
   storageClass: vsan-default-storage-policy
   tkr:
    reference:
     name: v1.25.7---vmware.3-fips.1-tkg.1
   vmClass: guaranteed-large
  nodePools:
  - name: worker-tkg-pool01
   replicas: 3
   storageClass: vsan-default-storage-policy
   tkr:
    reference:
     name: v1.25.7---vmware.3-fips.1-tkg.1
   vmClass: guaranteed-large
   volumes:
   - capacity:
     storage: 128Gi
    mountPath: /var/lib/containerd
    name: containerd
```

A TKR v1.26.5:

```
topology:
  controlPlane:
   replicas: 1
   storageClass: vsan-default-storage-policy
   tkr:
    reference:
     name: v1.26.5---vmware.2-fips.1-tkg.1
   vmClass: guaranteed-large
  nodePools:
  - name: worker-tkg-pool01
   replicas: 3
   storageClass: vsan-default-storage-policy
   tkr:
    reference:
     name: v1.26.5---vmware.2-fips.1-tkg.1
   vmClass: guaranteed-large
   volumes:
```

```
    - capacity:
       storage: 128Gi
     mountPath: /var/lib/containerd
     name: containerd
```

**Nota** Los nodos de plano de control y trabajo deben tener la misma versión de TKR. Puede actualizar todas las instancias de TKr o actualizar la versión del plano de control y eliminar el nombre de TKR de los nodos de trabajo.

Por ejemplo, para un clúster v1beta1, cambie de TKR v1.25.7:

```
apiVersion: cluster.x-k8s.io/v1beta1
  ...
  topology:
    class: tanzukubernetescluster
    version: v1.25.7---vmware.3-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      ...
    variables:
      ...
```

A TKR v1.26.5:

```
apiVersion: cluster.x-k8s.io/v1beta1
  ...
  topology:
    class: tanzukubernetescluster
    version: v1.26.5---vmware.2-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      ...
    variables:
      ...
```

7 Guarde los cambios que hizo en el archivo de manifiesto.

Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, el servicio de máquina virtual en el supervisor aprovisiona el nuevo nodo de trabajo.

8 Compruebe que kubectl notifique el correcto registro de los cambios en el manifiesto.

```
kubectl edit tanzukubernetescluster/tkg-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkg-cluster-1 edited
```

**Nota** Si recibe un error, o kubectl no informa de que el manifiesto del clúster se editó correctamente, asegúrese de haber configurado bien el editor de texto predeterminado con la variable de entorno KUBE_EDITOR. Consulte Configurar un editor de texto para Kubectl.

9 Compruebe que el clúster se esté actualizando.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                       AGE   PHASE
tkgs-cluster-1   3               3        v1.26.5---vmware.2-fips.1-tkg.1   21h   updating
```

10 Compruebe que el clúster se haya actualizado.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                       AGE   PHASE
tkgs-cluster-1   3               3        v1.26.5---vmware.2-fips.1-tkg.1    22h   running
```

# Actualizar un clúster de TKG mediante la edición de la clase de almacenamiento

Puede actualizar un clúster de TKG si cambia la clase de almacenamiento que utilizan los nodos del clúster.

Para iniciar una actualización gradual de un clúster de TKG, cambie el valor del parámetro `storageClass` en la especificación del clúster mediante el comando `kubectl edit`.

**Nota** No se puede usar el comando `kubectl apply` para actualizar un clúster de TKG implementado.

### Requisitos previos

Esta tarea requiere el uso del comando de edición kubectl. Este comando abre el manifiesto del clúster en el editor de texto definido por las variables de entorno KUBE_EDITOR o EDITOR. Al guardar el archivo, el clúster se actualiza con los cambios. Para configurar un editor para kubectl, consulte Configurar un editor de texto para Kubectl.

### Procedimiento

1 Realice la autenticación con Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de carga de trabajo de destino.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 Para determinar las clases de almacenamiento disponibles y decidir cuál debe usar, ejecute el siguiente comando.

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

4 Ejecute el siguiente comando para editar el manifiesto del clúster.

Clúster v1alpha3:

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

Clúster v1beta1:

```
kubectl edit cluster/CLUSTER-NAME
```

5 Edite el manifiesto cambiando el valor de `storageClass`.

Por ejemplo, para un clúster v1alpaha3, cambie el manifiesto del clúster de la clase `silver-storage-class` para los nodos de trabajo y el plano de control:

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: silver-storage-class
    workers:
      ...
      storageClass: silver-storage-class
```

Si desea usar la clase `gold-storage-class` para los nodos de trabajo y el plano de control:

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: gold-storage-class
    workers:
      ...
      storageClass: gold-storage-class
```

Del mismo modo, si aprovisionó un clúster v1beta1, actualice el valor de `variables.storageclass` en la especificación del clúster con el nombre de la clase de almacenamiento.

6 Guarde los cambios que hizo en el archivo de manifiesto.

Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, Tanzu Kubernetes Grid aprovisiona las máquinas virtuales del nuevo nodo y reduce la velocidad de las antiguas.

7 Compruebe que kubectl notifique el correcto registro de los cambios en el manifiesto.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

**Nota** Si recibe un error, o kubectl no informa de que el manifiesto del clúster se editó correctamente, asegúrese de haber configurado bien el editor de texto predeterminado con la variable de entorno KUBE_EDITOR. Consulte Configurar un editor de texto para Kubectl.

8 Compruebe que el clúster se haya actualizado.

Clúster v1alpha3:

```
kubectl get tanzukubernetescluster
```

Clúster v1beta1:

```
kubectl get cluster
```

# Actualizar un clúster de servicio TKG mediante la edición de la clase de Servicios

Puede actualizar un clúster de servicio TKG si cambia la clase de máquina virtual que se utiliza para alojar los nodos del clúster.

Para iniciar una actualización gradual de un clúster de servicio TKG, edite la definición de la `vmClass` mediante el comando `kubectl edit`. Se inician los nodos nuevos basados en la clase modificada y los nodos anteriores se desactivan.

**Nota** No se puede usar el comando `kubectl apply` para actualizar un clúster de TKG implementado.

#### Requisitos previos

Esta tarea requiere el uso del comando de edición kubectl. Este comando abre el manifiesto del clúster en el editor de texto definido por las variables de entorno KUBE_EDITOR o EDITOR. Al guardar el archivo, el clúster se actualiza con los cambios. Para configurar un editor para kubectl, consulte Configurar un editor de texto para Kubectl.

#### Procedimiento

1 Realice la autenticación con Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 Describa el clúster de TKG de destino y compruebe la clase de máquina virtual.

Clúster v1alpha3:

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

Clúster v1beta1:

```
kubectl describe cluster CLUSTER-NAME
```

4 Enumere y describa las clases de máquinas virtuales disponibles en el espacio de nombres de vSphere en el que se aprovisiona el clúster.

```
kubectl get virtualmachineclass
```

**Nota** La clase de VM de destino debe estar asociada con el espacio de nombres de vSphere en el que se aprovisionó el clúster de TKG. Consulte el servicio de TKG o la documentación del servicio de máquina virtual para obtener más información sobre el enlace de las clases de máquinas virtuales a espacios de nombres de vSphere.

5 Ejecute el siguiente comando para editar el manifiesto del clúster.

Clúster v1alpha3:

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

Clúster v1beta1:

```
kubectl edit cluster/CLUSTER-NAME
```

6 Edite el manifiesto cambiando la cadena de la clase de máquina virtual.

Por ejemplo, si utiliza un clúster v1alpah3, cambie el manifiesto del clúster para que no utilice la clase de máquina virtual `guaranteed-medium` para los nodos de trabajo:

```
topology:
   controlPlane:
     replicas: 3
     storageClass: vwk-storage-policy
     tkr:
       reference:
         name: v1.27.11---vmware.1-fips.1-tkg.2
     vmClass: guaranteed-medium
   nodePools:
   - name: worker-nodepool-a1
     replicas: 3
     storageClass: vwk-storage-policy
     tkr:
       reference:
         name: v1.27.11---vmware.1-fips.1-tkg.2
     vmClass: guaranteed-medium
```

Para usar la clase de máquina virtual `guaranteed-large` para los nodos de trabajo:

```
topology:
   controlPlane:
     replicas: 3
     storageClass: vwk-storage-policy
     tkr:
       reference:
         name: v1.27.11---vmware.1-fips.1-tkg.2
     vmClass: guaranteed-medium
   nodePools:
   - name: worker-nodepool-a1
     replicas: 3
     storageClass: vwk-storage-policy
     tkr:
       reference:
         name: v1.27.11---vmware.1-fips.1-tkg.2
     vmClass: guaranteed-large
```

Del mismo modo, si aprovisionó un clúster v1beta1, actualice el valor de `variables.vmclass` a la clase de máquina virtual de destino.

7 Guarde los cambios que hizo en el archivo de manifiesto.

Cuando guarde el archivo, kubectl aplicará los cambios al clúster. En segundo plano, el controlador de TKG aprovisiona las máquinas virtuales del nuevo nodo y reduce la velocidad de las antiguas.

8 Compruebe que kubectl notifique el correcto registro de los cambios en el manifiesto.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

**Nota** Si recibe un error, o kubectl no informa de que el manifiesto del clúster se editó correctamente, asegúrese de haber configurado bien el editor de texto predeterminado con la variable de entorno KUBE_EDITOR. Consulte Configurar un editor de texto para Kubectl.

9 Compruebe que el clúster se haya actualizado.

Clúster v1alpha3:

```
kubectl get tanzukubernetescluster
```

Clúster v1beta1:

```
kubectl get cluster
```

# Actualizar un clúster de TKG mediante la CLI de Tanzu

Actualice un clúster de TKG actualizando la versión de la versión de Tanzu Kubernetes mediante la CLI de Tanzu.

Para iniciar una actualización gradual de un clúster de TKGS, actualice la versión de la versión de Tanzu Kubernetes mediante la CLI de Tanzu.

Consulte la *Guía de referencia de la CLI de Tanzu* para ver todos los detalles de uso.

### Requisitos previos

Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG.

### Procedimiento

1 Realice la autenticación con Supervisor.

2 Enumere el clúster de TKG.

```
tanzu cluster list
```

3 Actualice el clúster de TKG.

```
tanzu cluster upgrade CLUSTER-NAME --tkr TKR-NAME -n VSPHERE-NAMESPACE
```

Donde:

- `CLUSTER-NAME` es el nombre del clúster de TKG en el que desea realizar la actualización
- `TKR-NAME` es la cadena de la versión de TKR
- `VSPHERE-NAMESPACE` es el nombre del espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG

Por ejemplo:

```
tanzu cluster upgrade tkg-cluster-1 --tkr v1.23.8---vmware.2-tkg.2-zshippable -n tkg2-cluster-ns
```

4 Confirme la actualización del clúster.

Cuando se actualice el clúster, verá un mensaje similar al siguiente:

```
Cluster 'tkg-cluster-1' successfully upgraded to kubernetes version 'v1.23.8+vmware.2-tkg.2-zshippable'
```

# Ajuste de escala automático de clústeres de servicio TKG

10

En esta sección se proporciona información sobre el ajuste de escala automático de clústeres de servicio TKG.

Lea los siguientes temas a continuación:

- Acerca del ajuste de escala automático de clústeres
- Instalar el escalador automático de clústeres mediante Kubectl
- Instalar el escalador automático de clústeres mediante la CLI de Tanzu
- Actualizar clúster de escalado automático mediante Kubectl
- Actualizar clúster de escalado automático mediante la CLI de Tanzu
- Prueba del escalador automático de clústeres
- Eliminar el escalador automático de clústeres

## Acerca del ajuste de escala automático de clústeres

Puede implementar el escalador automático de clústeres de TKG para ajustar automáticamente el número de nodos de trabajo en un clúster de servicio TKG en función de las demandas de las cargas de trabajo.

### Acerca del ajuste de escala automático de clústeres

El escalador automático de clústeres de servicio TKG es una implementación del escalador automático del clúster de Kubernetes. Para obtener más información, consulte la documentación del escalador automático de clústeres.

El escalador automático de clústeres admite el escalado horizontal y el escalado vertical de los nodos del clúster. Si ejecuta el clúster en una instancia de Supervisor de varias zonas, el escalador automático puede ampliar los grupos de nodos asignados a una zona de disponibilidad específica.

El escalador automático de clústeres se proporciona como un paquete estándar que se instala en el clúster mediante Kubectl o la CLI de Tanzu. El escalador automático de clústeres se ejecuta como una implementación en el clúster de TKG con las credenciales de la cuenta de servicio.

Existe una relación 1 a 1 entre la versión secundaria del paquete del escalador automático y la versión secundaria de TKr. Por ejemplo, si utiliza TKr 1.27.11, debe instalar la versión 1.27.2 del escalador automático. Si la versión no coincide, se producirá un error en la reconciliación del paquete.

Si bien el escalador automático de clústeres admite tanto el escalado horizontal como el escalado vertical de los nodos de trabajo, hay algunos casos en los que el escalador automático de clústeres no reduce el escalado vertical de los nodos, ya que algunos tipos de aplicaciones impiden que los nodos se reduzcan. Consulte "¿Qué tipos de pods pueden evitar que CA elimine un nodo?" en la documentación del escalador automático de clústeres.

## Requisitos de la versión

El escalador automático de clústeres tiene los siguientes requisitos de versión.

- La versión mínima de vSphere es vSphere 8 U3.
- La versión mínima de TKr es TKr 1.27.x para vSphere 8.
- La versión secundaria de TKr y la versión secundaria del paquete del escalador automático de clústeres deben coincidir.

## Requisitos del paquete

El escalador automático de clústeres se proporciona como un paquete estándar. La versión secundaria del paquete debe coincidir con la versión secundaria del TKr que se está utilizando. Por ejemplo, si utiliza TKr 1.27.11, debe instalar la versión 1.27.2 del escalador automático. Si la versión no coincide, se producirá un error en la reconciliación del paquete.

Es posible que deba localizar el paquete de destino en una versión de repositorio posterior. Por ejemplo, la versión 1.27.2 del escalador automático se encuentra en la versión 2024.4.12 del repositorio de paquetes estándar. Las versiones posteriores del paquete del escalador automático, como 1.28.x, 1.29.x, 1.30.x, etc., se encuentran en las versiones del repositorio posteriores. Para acceder a todos los repositorios de paquetes estándar, ejecute el siguiente comando:

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/repo
```

## Flujo de trabajo

El flujo de trabajo de alto nivel para habilitar el ajuste de escala automático de clústeres es el siguiente:

1 Cree un nuevo clúster de TKG o actualice uno existente con anotaciones de escalador automático y elimine el campo de réplicas en `spec.topology.workers.machinedeployments`.

2 Instale el repositorio de paquetes en el clúster de TKG que creó o actualizó.

3 Instale el paquete de escalador automático en el clúster de TKG que creó o actualizó.

El escalador automático se instala en el clúster de TKG como una implementación en el espacio de nombres kube-system.

Consulte los siguientes temas para obtener instrucciones detalladas:

- Instalar el escalador automático de clústeres mediante Kubectl
- Instalar el escalador automático de clústeres mediante la CLI de Tanzu

# Instalar el escalador automático de clústeres mediante Kubectl

Consulte estas instrucciones para instalar y configurar el paquete del escalador automático de clústeres mediante kubectl.

## Requisitos

Cumpla con los siguientes requisitos.

- La versión mínima de vSphere es vSphere 8 U3.
- La versión mínima de TKr es TKr 1.27.x para vSphere 8.
- La versión secundaria de TKr y la versión secundaria del paquete del escalador automático de clústeres deben coincidir.

**Atención** Existe una relación 1 a 1 entre la versión secundaria del paquete del escalador automático y la versión secundaria de TKr. Por ejemplo, si utiliza TKr 1.27.11, debe instalar la versión 1.27.2 del escalador automático. Si la versión no coincide, se producirá un error en la reconciliación del paquete.

## Configurar espacio de nombres de vSphere

Complete las siguientes tareas de requisitos previos para el aprovisionamiento del clúster de TKG.

1. Instale o actualice el entorno a vSphere 8 U3 y TKr 1.27.x para vSphere 8.
2. Cree o actualice una biblioteca de contenido con las versiones de Tanzu Kubernetes más recientes. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.
3. Cree y configure un espacio de nombres de vSphere para alojar el clúster de TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.
4. Instale Herramientas de la CLI de Kubernetes para vSphere.

El siguiente ejemplo se puede utilizar para instalar las herramientas desde la línea de comandos. Para obtener más instrucciones, consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

```
curl -LOk https://${SUPERVISOR_IP-or-FQDN}/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

5 Ejecute `kubectl` y `kubectl vsphere` para comprobar la instalación.

## Crear un clúster de TKG con anotaciones del escalador automático

Siga las instrucciones para crear un clúster de TKG. Para obtener más instrucciones, consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

Para utilizar el escalador automático, debe configurar el clúster con anotaciones en la etiqueta del escalador automático, como se muestra en el ejemplo de especificación de clúster que se proporciona aquí. A diferencia del aprovisionamiento de clústeres normal, no codifica de forma forzada el número de réplicas del nodo de trabajo. Kubernetes tiene una lógica predeterminada integrada para las réplicas en función de las anotaciones de tamaño máximo y mínimo del escalador automático. Dado que se trata de un clúster nuevo, se utiliza el tamaño mínimo para crear el clúster. Para obtener más información, consulte https://cluster-api.sigs.k8s.io/tasks/automated-machine-management/autoscaling.

1 Autentíquese con Supervisor mediante kubectl.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN --vsphere-
username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere de destino que alojará el clúster.

```
kubectl config use-context tkgs-cluster-namespace
```

3 Enumere las clases de máquina virtual que están disponibles en espacio de nombres de vSphere.

   Solo puede utilizar las clases de máquinas virtuales que están enlazadas al espacio de nombres de vSphere de destino. Consulte Uso de clases de máquinas virtuales con clústeres de Servicio TKG.

4 Mencione las clases de almacenamiento de volumen persistente disponibles.

```
kubectl describe namespace VSPHERE-NAMESPACE-NAME
```

   El comando devuelve detalles sobre el espacio de nombres de vSphere, incluida la clase de almacenamiento. El comando `kubectl describe storageclasses` también devuelve las clases de almacenamiento disponibles, pero requiere permisos de administrador de vSphere.

5 Lista de versiones de Tanzu Kubernetes disponibles:

```
kubectl get tkr
```

Este comando devuelve los TKr disponibles en este espacio de nombres de vSphere y su compatibilidad. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

6 Utilice la información que recopiló para crear un archivo YAML de especificación de clúster de TKG con la configuración del escalador automático de clústeres requerida.

- Utilice las anotaciones `*-min-size` y `*-max-size` para nodePools de trabajo. En este ejemplo, 3 es el mínimo y 5 es el número máximo de nodos de trabajo que se pueden escalar. De forma predeterminada, el clúster se creará con 3 nodos de trabajo.
- Utilice la versión secundaria coincidente para el paquete de TKr y el escalador automático.
- Los valores de `metadata.name` y `metadata.namespace` del clúster utilizados son coherentes con los valores predeterminados del paquete del escalador automático. Si cambia estos valores en la especificación del clúster, deberá modificarlos en `autoscaler-data-values` (consulte a continuación).

```
#cc-autoscaler.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
 name: gc1
 namespace: cluster
spec:
 clusterNetwork:
   pods:
     cidrBlocks:
     - 192.0.2.0/16
   serviceDomain: cluster.local
   services:
     cidrBlocks:
     - 198.51.100.0/12
 topology:
   class: tanzukubernetescluster
   controlPlane:
     metadata: {}
     replicas: 3
   variables:
   - name: storageClasses
     value:
     - wcpglobal-storage-profile
   - name: vmClass
     value: guaranteed-medium
   - name: storageClass
     value: wcpglobal-storage-profile
   #minor versions must match
   version: v1.27.11---vmware.1-fips.1-tkg.2
```

```
    workers:
      machineDeployments:
      - class: node-pool
        metadata:
          annotations:
            cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size: "3"
            cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size: "5"
        name: np-1
```

7 Aplique la especificación del clúster.

```
kubectl apply -f cc-autoscaler.yaml
```

8 Compruebe la creación del clúster.

```
kubectl get cluster,vm
```

9 Compruebe la versión del nodo del clúster.

```
kubectl get node
```

## Instalar el administrador de paquetes en el clúster de TKG

Una vez que se aprovisione el clúster de TKG, instale el administrador de paquetes en el clúster y configure el repositorio de paquetes.

1 Inicie sesión en el clúster de TKG que aprovisionó.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN \
--vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

2 Instale la herramienta Carvel imgpkg.

```
wget -O- https://carvel.dev/install.sh > install.sh
sudo bash install.sh
```

3 Ejecute `imgpkg version` para comprobar la instalación.

4 Compruebe la versión del repositorio de paquetes.

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/repo
```

5 Instale el repositorio de paquetes. Actualice la versión del repositorio según corresponda.

```
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageRepository
metadata:
  name: tanzu-standard
  namespace: tkg-system
```

```
spec:
  fetch:
    imgpkgBundle:
      image: projects.registry.vmware.com/tkg/packages/standard/repo:v2024.4.12
```

6 Compruebe el repositorio de paquetes.

```
kubectl get packagerepository -A
NAMESPACE    NAME              AGE     DESCRIPTION
tkg-system   tanzu-standard   2m22s   Reconcile succeeded
```

7 Compruebe la existencia del paquete del escalador automático de clústeres.

```
kubectl get package
NAME                                                                  PACKAGEMETADATA
NAME                             VERSION                 AGE
cert-manager.tanzu.vmware.com.1.7.2+vmware.3-tkg.1                    cert-
manager.tanzu.vmware.com                    1.7.2+vmware.3-tkg.1    5s
cert-manager.tanzu.vmware.com.1.7.2+vmware.3-tkg.3                    cert-
manager.tanzu.vmware.com                    1.7.2+vmware.3-tkg.3    5s
cluster-autoscaler.tanzu.vmware.com.1.25.1+vmware.1-tkg.3             cluster-
autoscaler.tanzu.vmware.com              1.25.1+vmware.1-tkg.3   5s
cluster-autoscaler.tanzu.vmware.com.1.26.2+vmware.1-tkg.3             cluster-
autoscaler.tanzu.vmware.com              1.26.2+vmware.1-tkg.3   5s
cluster-autoscaler.tanzu.vmware.com.1.27.2+vmware.1-tkg.3             cluster-
autoscaler.tanzu.vmware.com              1.27.2+vmware.1-tkg.3   5s
contour.tanzu.vmware.com.1.26.2+vmware.1-tkg.1
contour.tanzu.vmware.com                         1.26.2+vmware.1-tkg.1   5s
...
```

## Instalar el paquete del escalador automático

Ahora puede instalar el paquete del escalador automático de clústeres. El escalador automático de clústeres se instalará como una implementación en el espacio de nombres `kube-system`.

1 Cree el archivo de configuración `autoscaler.yaml`.

- Para personalizar el escalador automático, cambie la sección `autoscaler-data-values` de la especificación por los valores adecuados para su entorno.

```
#autoscaler.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: autoscaler-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: autoscaler-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: autoscaler-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: autoscaler
  namespace: tkg-system
spec:
  serviceAccountName: autoscaler-sa
  packageRef:
    refName: cluster-autoscaler.tanzu.vmware.com
    versionSelection:
      constraints: 1.27.2+vmware.1-tkg.3
  values:
  - secretRef:
      name: autoscaler-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: autoscaler-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    arguments:
      ignoreDaemonsetsUtilization: true
      maxNodeProvisionTime: 15m
      maxNodesTotal: 0
      metricsPort: 8085
      scaleDownDelayAfterAdd: 10m
      scaleDownDelayAfterDelete: 10s
      scaleDownDelayAfterFailure: 3m
      scaleDownUnneededTime: 10m
    clusterConfig:
      clusterName: "gc1"
      clusterNamespace: "cluster"
    paused: false
```

2 Instale el paquete del escalador automático de clústeres.

```
kubectl apply -f autoscaler.yaml
```

3 Compruebe la instalación del paquete del escalador automático.

```
kubectl get pkgi -A | grep autoscaler
```

Resultado esperado:

```
tkg-system autoscaler cluster-autoscaler.tanzu.vmware.com 1.27.2+vmware.1-tkg.3 Reconcile
succeeded 3m52s
```

4 Compruebe la implementación del escalador automático.

```
kubectl get pods -n kube-system | grep autoscaler
```

```
cluster-autoscaler-798b65bd9f-bht8n 1/1 Running 0 2m
```

## Probar el ajuste de escala automático de clústeres

Para probar el ajuste de escala automático de clústeres, implemente una aplicación, aumente el número de réplicas y compruebe que los nodos de trabajo adicionales se hayan escalado para controlar la carga.

Consulte Prueba del escalador automático de clústeres.

## Actualizar un clúster de ajuste de escala automático

Para actualizar un clúster con ajuste de escala automático, ponga en pausa el paquete del escalador automático.

Consulte Actualizar clúster de escalado automático mediante Kubectl.

# Instalar el escalador automático de clústeres mediante la CLI de Tanzu

Consulte estas instrucciones para instalar y configurar el paquete del escalador automático de clústeres mediante la CLI de Tanzu.

## Requisitos

Cumpla con los siguientes requisitos.

- La versión mínima de vSphere es vSphere 8 U3, incluidos los hosts ESXi y vCenter
- La versión mínima de TKr es TKr 1.27.x para vSphere 8.
- La versión secundaria de TKr y la versión secundaria del paquete del escalador automático de clústeres deben coincidir.

**Nota** Existe una relación 1 a 1 entre la versión secundaria del paquete del escalador automático y la versión secundaria de TKr. Por ejemplo, si utiliza TKr 1.27.11, debe instalar la versión 1.27.2 del escalador automático. Si la versión no coincide, se producirá un error en la reconciliación del paquete.

## Configurar espacio de nombres de vSphere

Complete las siguientes tareas de requisitos previos para el aprovisionamiento del clúster de TKG.

1 Instale o actualice el entorno a vSphere 8 U3 y TKr 1.27.x para vSphere 8.

2 Cree o actualice una biblioteca de contenido con las versiones de Tanzu Kubernetes más recientes. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

3 Cree y configure un espacio de nombres de vSphere para alojar el clúster de TKG. Consulte Capítulo 6 Configurar espacios de nombres de vSphere para alojar clústeres de Servicio TKG.

4 Instale Herramientas de la CLI de Kubernetes para vSphere.

   El siguiente ejemplo se puede utilizar para instalar las herramientas desde la línea de comandos. Para obtener más instrucciones, consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

```
wget https://SUPERVISOR-IP-or-FQDN/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
chmod +x bin/kubectl*
mv bin/kubectl* /usr/bin/kubectl vsphere --help
rm ~/.kube/config
kubectl vsphere login --insecure-skip-tls-verify --server SUPERVISOR-IP-or-FQDN --tanzu-
kubernetes-cluster-namespace VSPHERE-NAMESPACE --vsphere-username VSPHERE-USER
kubectl config use-context VSPHERE-NAMESPACE
```

5 Ejecute `kubectl` y `kubectl vsphere` para comprobar la instalación.

## Crear un clúster de TKG con anotaciones del escalador automático

Siga las instrucciones para crear un clúster de TKG. Para obtener más instrucciones, consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

Para utilizar el escalador automático, debe configurar el clúster con anotaciones en la etiqueta del escalador automático, como se muestra en el ejemplo de especificación de clúster que se proporciona aquí. A diferencia del aprovisionamiento de clústeres normal, no codifica de forma forzada el número de réplicas del nodo de trabajo. Kubernetes tiene una lógica predeterminada integrada para las réplicas en función de las anotaciones de tamaño máximo y mínimo del escalador automático. Dado que se trata de un clúster nuevo, se utiliza el tamaño mínimo para crear el clúster. Para obtener más información, consulte https://cluster-api.sigs.k8s.io/tasks/automated-machine-management/autoscaling.

1 Autentíquese con Supervisor mediante kubectl.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN --vsphere-
username USERNAME
```

2 Cambie el contexto al espacio de nombres de vSphere de destino que alojará el clúster.

```
kubectl config use-context tkgs-cluster-namespace
```

3 Enumere las clases de máquina virtual que están disponibles en espacio de nombres de vSphere.

Solo puede utilizar las clases de máquinas virtuales que están enlazadas al espacio de nombres de vSphere de destino. Consulte Uso de clases de máquinas virtuales con clústeres de Servicio TKG.

4 Mencione las clases de almacenamiento de volumen persistente disponibles.

```
kubectl describe namespace VSPHERE-NAMESPACE-NAME
```

El comando devuelve detalles sobre el espacio de nombres de vSphere, incluida la clase de almacenamiento. El comando `kubectl describe storageclasses` también devuelve las clases de almacenamiento disponibles, pero requiere permisos de administrador de vSphere.

5 Lista de versiones de Tanzu Kubernetes disponibles:

```
kubectl get tkr
```

Este comando devuelve los TKr disponibles en este espacio de nombres de vSphere y su compatibilidad. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

6 Utilice la información que recopiló para crear un archivo YAML de especificación de clúster de TKG con la configuración del escalador automático de clústeres requerida.

- Utilice las anotaciones `*-min-size` y `*-max-size` para nodePools de trabajo. En este ejemplo, 3 es el mínimo y 5 es el número máximo de nodos de trabajo que se pueden escalar. De forma predeterminada, el clúster se creará con 3 nodos de trabajo.
- Utilice la versión secundaria coincidente para el paquete de TKr y el escalador automático.
- Los valores de `metadata.name` y `metadata.namespace` del clúster utilizados son coherentes con los valores predeterminados del paquete del escalador automático. Si cambia estos valores en la especificación del clúster, deberá modificarlos en `autoscaler-data-values` (consulte a continuación).

```
#cc-autoscaler.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
 name: tkc
 namespace: cluster
spec:
 clusterNetwork:
   pods:
     cidrBlocks:
```

```
      - 192.0.2.0/16
    serviceDomain: cluster.local
    services:
      cidrBlocks:
      - 198.51.100.0/12
  topology:
    class: tanzukubernetescluster
    controlPlane:
      metadata: {}
      replicas: 3
    variables:
    - name: storageClasses
      value:
      - wcpglobal-storage-profile
    - name: vmClass
      value: guaranteed-medium
    - name: storageClass
      value: wcpglobal-storage-profile
    #minor versions must match
    version: v1.27.11---vmware.1-fips.1-tkg.2
    workers:
      machineDeployments:
      - class: node-pool
        metadata:
          annotations:
            cluster.x-k8s.io/cluster-api-autoscaler-node-group-min-size: "3"
            cluster.x-k8s.io/cluster-api-autoscaler-node-group-max-size: "5"
        name: np-1
```

7 Aplique la especificación del clúster.

```
kubectl apply -f cc-autoscaler.yaml
```

8 Compruebe la creación del clúster.

```
kubectl get cluster,vm
```

9 Compruebe la versión del nodo del clúster.

```
kubectl get node
```

## Crear el repositorio de paquetes en el clúster de TKG

Una vez que se aprovisione el clúster de TKG, instale la CLI de Tanzu y configure el repositorio de paquetes.

1 Instale la CLI de Tanzu.

Consulte Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG.

2 Inicie sesión en el clúster.

```
rm ~/.kube/config
kubectl vsphere login --insecure-skip-tls-verify --server 192.168.0.2 --tanzu-kubernetes-cluster-namespace autoscaler --vsphere-username administrator@vsphere.local --tanzu-kubernetes-cluster-name cckubectl
config use-context cc
```

3 Cree el repositorio de paquetes.

```
#Standard package repository URL might change depending on the required cluster autoscaler version
tanzu package repository add standard-repo --url projects.registry.vmware.com/tkg/packages/standard/repo:v2024.4.12 -n tkg-system
tanzu package available list -n tkg-system
tanzu package available get cluster-autoscaler.tanzu.vmware.com -n tkg-system
```

## Instalar el paquete del escalador automático

Instale el paquete del escalador automático de clústeres. El escalador automático de clústeres se instalará en el espacio de nombres `kube-system`.

1 Genere el `values.yaml` predeterminado mediante el comando de la CLI de Tanzu.

```
tanzu package available get cluster-autoscaler.tanzu.vmware.com/1.27.2+vmware.1-tkg.3  -n tkg-system --default-values-file-output values.yaml
```

2 Actualice el `values.yaml` para la instalación del paquete.

```
arguments:
  ignoreDaemonsetsUtilization: true
  maxNodeProvisionTime: 15m
  maxNodesTotal: 0
  metricsPort: 8085
  scaleDownDelayAfterAdd: 10m
  scaleDownDelayAfterDelete: 10s
  scaleDownDelayAfterFailure: 3m
  scaleDownUnneededTime: 10m
clusterConfig:
  clusterName: "tkc"
  clusterNamespace: "cluster"
paused: false
```

3 Instale el paquete del escalador automático de clústeres mediante la CLI de Tanzu.

```
tanzu package install cluster-autoscaler-pkgi -n tkg-system --package cluster-autoscaler.tanzu.vmware.com --version 1.27.2+vmware.1-tkg.3 --values-file values.yaml
```

## Probar el ajuste de escala automático de clústeres

Para probar el ajuste de escala automático de clústeres, implemente una aplicación, aumente el número de réplicas y compruebe que se escalen los nodos de trabajo adicionales para gestionar la carga adicional.

Consulte Prueba del escalador automático de clústeres.

## Actualizar un clúster de ajuste de escala automático

Para actualizar un clúster de ajuste de escala automático, primero debe pausar el paquete del escalador automático.

Consulte Actualizar clúster de escalado automático mediante la CLI de Tanzu.

# Actualizar clúster de escalado automático mediante Kubectl

Antes de actualizar un clúster de TKG, el escalador automático debe estar en pausa. Después de actualizar la versión de TKr del clúster, debe actualizar la versión del paquete del escalador automático para que coincida con la versión secundaria de TKr.

## Requisitos

En esta tarea, se supone que se instaló el escalador automático de clústeres en un clúster de TKG. Consulte Instalar el escalador automático de clústeres mediante Kubectl.

## Antes de actualizar el clúster: poner en pausa el escalador automático

Antes de actualizar un clúster de TKG con el escalador automático instalado, primero debe poner en pausa el paquete del escalador automático.

1 Ponga en pausa el paquete del escalador automático de clústeres estableciendo el valor booleano `paused` en `true` en el secreto de `autoscaler-data-values.yaml`.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: autoscaler-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    arguments:
      ignoreDaemonsetsUtilization: true
      maxNodeProvisionTime: 15m
      maxNodesTotal: 0
      metricsPort: 8085
      scaleDownDelayAfterAdd: 10m
      scaleDownDelayAfterDelete: 10s
```

```
      scaleDownDelayAfterFailure: 3m
      scaleDownUnneededTime: 10m
    clusterConfig:
      clusterName: "gc1"
      clusterNamespace: "cluster"
    paused: true
```

2 Aplique las actualizaciones al secreto de `autoscaler-data-values`.

```
kubectl apply -f autoscaler-data-values.yaml
```

## Actualizar el clúster

Una vez que el escalador automático esté en pausa, continúe con la actualización del clúster.

1 Actualice la versión de Kubernetes del clúster de TKG.

   Consulte Actualizar un clúster de TKG mediante el cambio de la versión de TKR.

## Después de actualizar el clúster: actualizar la versión del paquete del escalador automático

Después de actualizar el clúster, actualice la versión del paquete del escalador automático para que coincida con la versión secundaria de TKr y deshabilite la pausa.

1 Elija la versión del escalador automático correspondiente.

   Las versiones secundarias de TKr y del paquete del escalador automático deben coincidir. Por ejemplo, si actualizó el clúster a TKr v1.28.8, deberá utilizar el paquete del escalador automático v1.28.x.

2 Para actualizar los recursos del escalador automático, establezca la versión del escalador automático de destino y restablezca la pausa a false.

```
#autoscaler-package-upgrade.yaml
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: autoscaler
  namespace: tkg-system
spec:
  serviceAccountName: autoscaler-sa
  packageRef:
    refName: cluster-autoscaler.tanzu.vmware.com
    versionSelection:
      constraints: 1.28.0+vmware.1-tkg.1
  values:
  - secretRef:
      name: autoscaler-data-values
---
apiVersion: v1
kind: Secret
metadata:
```

```
  name: autoscaler-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    arguments:
      ignoreDaemonsetsUtilization: true
      maxNodeProvisionTime: 15m
      maxNodesTotal: 0
      metricsPort: 8085
      scaleDownDelayAfterAdd: 10m
      scaleDownDelayAfterDelete: 10s
      scaleDownDelayAfterFailure: 3m
      scaleDownUnneededTime: 10m
    clusterConfig:
      clusterName: "gc1"
      clusterNamespace: "cluster"
    paused: false
```

3 Aplique las actualizaciones al paquete del escalador automático.

```
kubectl apply -f autoscaler-package-upgrade.yaml
```

4 Compruebe que el pod del escalador automático se ejecuta en el espacio de nombres kube-system.

5 Pruebe el escalador automático de clústeres.

Prueba del escalador automático de clústeres.

# Actualizar clúster de escalado automático mediante la CLI de Tanzu

Antes de actualizar un clúster de TKG, el escalador automático debe estar en pausa. Después de actualizar la versión de TKr del clúster, debe actualizar la versión del paquete del escalador automático para que coincida con la versión secundaria de TKr.

## Requisitos

En esta tarea, se supone que se instaló el escalador automático de clústeres en un clúster de TKG. Consulte Instalar el escalador automático de clústeres mediante la CLI de Tanzu.

## Antes de actualizar el clúster: poner en pausa el escalador automático

Antes de actualizar un clúster de TKG con el escalador automático instalado, primero debe poner en pausa el paquete del escalador automático.

1 Para poner en pausa el paquete del escalador automático de clústeres, establezca el valor booleano `paused` en `true` en el archivo de configuración `values.yaml`.

```
arguments:
  ignoreDaemonsetsUtilization: true
  maxNodeProvisionTime: 15m
  maxNodesTotal: 0
  metricsPort: 8085
  scaleDownDelayAfterAdd: 10m
  scaleDownDelayAfterDelete: 10s
  scaleDownDelayAfterFailure: 3m
  scaleDownUnneededTime: 10m
clusterConfig:
  clusterName: "tkc"
  clusterNamespace: "cluster"
paused: true #set to true before upgrade
```

2 Actualice el paquete mediante la CLI de Tanzu.

```
tanzu package installed update cluster-autoscaler-pkgi -n tkg-system --package cluster-autoscaler.tanzu.vmware.com --values-file values.yaml
```

## Actualizar el clúster

Una vez que el escalador automático esté en pausa, continúe con la actualización del clúster.

1 Actualice la versión de Kubernetes del clúster de TKG.

Consulte Actualizar un clúster de TKG mediante el cambio de la versión de TKR.

## Después de actualizar el clúster: actualizar la versión del paquete del escalador automático

Después de actualizar el clúster, actualice la versión del paquete del escalador automático para que coincida con la versión secundaria de la TKr y restablezca a false la clave que puso en pausa.

1 Elija la versión del escalador automático correspondiente.

Las versiones secundarias de TKr y del paquete del escalador automático deben coincidir. Por ejemplo, si actualizó el clúster a TKr v1.28.8, deberá utilizar el paquete del escalador automático v1.28.0.

2 Genere el `values.yaml` predeterminado mediante el comando de la CLI de Tanzu.

```
tanzu package available get cluster-autoscaler.tanzu.vmware.com/1.28.0+vmware.1-tkg.1  -n tkg-system --default-values-file-output new-values.yaml
```

3 Actualice el archivo `new-values.yaml` con la nueva versión del paquete y restablezca a false la clave que puso en pausa.

4 Utilice la CLI de Tanzu para actualizar la instalación del escalador automático de clústeres.

```
tanzu package installed update cluster-autoscaler-pkgi -n tkg-system --package cluster-
autoscaler.tanzu.vmware.com --values-file new-values.yaml --version 1.28.1+vmware.1-tkg.1
```

# Prueba del escalador automático de clústeres

Consulte estas instrucciones para probar un escalador automático de clústeres instalado.

## Requisitos

En esta tarea, se supone que se instaló el escalador automático de clústeres en un clúster de TKG.

- Instalar el escalador automático de clústeres mediante Kubectl
- Instalar el escalador automático de clústeres mediante la CLI de Tanzu

## Prueba del escalador automático de clústeres

Para comprobar que el escalador automático escala automáticamente los nodos de trabajo, implemente una aplicación y después escale el número de réplicas en la implementación. El escalador automático escalará verticalmente los nodos de trabajo una vez que los recursos del nodo sean insuficientes.

1 Cree la siguiente definición de aplicación denominada `app.yaml`.

```
apiVersion: v1
kind: Namespace
metadata:
  name: app
  labels:
    pod-security.kubernetes.io/enforce: privileged
---
apiVersion: v1
kind: Service
metadata:
  name: application-cpu
  namespace: app
  labels:
    app: application-cpu
spec:
  type: ClusterIP
  selector:
    app: application-cpu
  ports:
    - protocol: TCP
      name: http
      port: 80
```

```
      targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: application-cpu
  namespace: app
  labels:    app: application-cpu
spec:
  selector:
    matchLabels:
      app: application-cpu
    replicas: 1
    strategy:
      type: RollingUpdate
      rollingUpdate:
        maxSurge: 1
        maxUnavailable: 0
    template:
      metadata:
        labels:
          app: application-cpu
      spec:
        containers:
        - name: application-cpu
          image: wcp-docker-ci.artifactory.eng.vmware.com/app-cpu:v1.0.0
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          resources:
            requests:
              memory: 50Mi
              cpu: 500m
            limits:
              memory: 500Mi
              cpu: 2000m
```

2 Cree la aplicación.

```
kubectl apply -f app.yaml
```

3 Escale verticalmente las réplicas de la aplicación para activar el escalador automático.

Por ejemplo, aumente el número de `spec.selector.replicas` de 1 a un número mayor para que se necesiten nodos de trabajo adicionales.

4 Actualice la aplicación.

```
kubectl apply -f app.yaml
```

5 Compruebe que se crearon nodos de trabajo adicionales para gestionar la carga.

El escalador automático escalará verticalmente el número de nodos de trabajo una vez que los recursos del nodo sean insuficientes.

# Eliminar el escalador automático de clústeres

Consulte estas instrucciones para eliminar un escalador automático de clústeres instalado.

## Requisitos

En esta tarea, se supone que se instaló el escalador automático de clústeres en un clúster de TKG.

- Instalar el escalador automático de clústeres mediante Kubectl
- Instalar el escalador automático de clústeres mediante la CLI de Tanzu

## Eliminar el escalador automático de clústeres mediante Kubectl

1 Para eliminar el escalador automático de clústeres mediante Kubectl, utilice el siguiente comando.

```
kubectl delete -f autoscaler.yaml
```

**Nota** El nombre `autoscaler.yaml` es el nombre que utilizó al implementar el paquete del escalador automático. Si utiliza un nombre diferente, actualice el comando según corresponda. Consulte Instalar el escalador automático de clústeres mediante Kubectl.

## Eliminar el escalador automático de clústeres mediante la CLI de Tanzu

1 Para eliminar el escalador automático de clústeres mediante la CLI de Tanzu, utilice el siguiente comando.

```
tanzu package installed delete -n tkg-system cluster-autoscaler-pkgi
```

# Instalar paquetes estándar en clústeres de Servicio TKG

11

VMware proporciona un conjunto estándar de aplicaciones de código abierto agrupadas como paquetes que se pueden instalar en sus clústeres de Servicio TKG para realizar operaciones.

Lea los siguientes temas a continuación:



## Paquetes estándar para clústeres de Servicio TKG

El vSphere IaaS control plane admite paquetes estándar para la instalación en clústeres de Servicio TKG.

### Paquetes compatibles para clústeres de Servicio TKG

En la tabla se enumeran los paquetes estándar disponibles para la instalación en clústeres de Servicio TKG aprovisionados con TKr para vSphere 8.x. Complete los requisitos previos necesarios antes de instalar el paquete de destino.

| Paquete | Descripción | Instrucciones |
| --- | --- | --- |
| Administrador de certificados | Administración de certificados | Instalar administrador de certificados |
| Contour con Envoy | Controladora de entrada de Kubernetes y proxy inverso | Instalar Contour con Envoy |
| ExternalDNS | Búsqueda de DNS de los servicios de Kubernetes | Instalar ExternalDNS |
| Fluent Bit | Reenvío de registros | Instalar ExternalDNS |
| Prometheus con Alertmanager | Supervisión y alertas | Instalar Prometheus con Alertmanager |
| Grafana | Visualización | Instalar Grafana |
| Harbor | Registro de contenedores | Instalar Registro de Harbor |

| Paquete | Descripción | Instrucciones |
|---|---|---|
| Webhook de validación de instantáneas de CSI externa<br>Webhook de CSI de PV de vSphere | Creación de instantáneas de almacenamiento persistente | Capítulo 15 Crear instantáneas en un clúster de Servicio TKG |
| Escalador automático de clústeres | Escalado automático de clústeres | Capítulo 10 Ajuste de escala automático de clústeres de servicio TKG |

# Instalar paquetes estándar en un clúster de TKG mediante TKr para vSphere 8.x

Consulte esta sección para instalar los paquetes estándar en un clúster de Servicio TKG aprovisionado con TKr para vSphere 8.x.

## Requisitos generales

Cumpla con estos requisitos generales para instalar los paquetes estándar en un clúster de Servicio TKG.

### Requisitos de plataforma

Estas instrucciones son específicas para instalar paquetes estándar en clústeres de TKG aprovisionados con TKr para vSphere 8.x. Consulte las notas de la versión de TKr para obtener más información.

Si va a implementar paquetes estándar en clústeres de TKG aprovisionados con un TKr para vSphere 7.x, consulte Instalar paquetes estándar en un clúster de TKG mediante TKr para vSphere 7.x.

### Requisitos de repositorio

vSphere IaaS control plane admite la instalación de paquetes estándar en clústeres de TKG para las TKr compatibles con vSphere 8. La Uso de versiones de Kubernetes con clústeres de Servicio TKG de vSphere 8 incluye el sistema de empaquetado Carvel y la controladora Kapp. Estos componentes se administran automáticamente como parte de la imagen de TKr en la que se basan los nodos de TKG. Consulte las Notas de la versión de TKR para conocer los detalles de la compatibilidad de TKr con vSphere.

### Requisitos del cliente

La instalación de paquetes estándar en clústeres de TKG aprovisionados con TKr para vSphere 8.x requiere Herramientas de la CLI de Kubernetes para vSphere, incluidos Kubectl, complemento de vSphere para kubectl y CLI de Tanzu. Para instalar estas herramientas, consulte Instalar las herramientas de CLI para clústeres de Servicio TKG.

## Requisitos de almacenamiento

El clúster de TKG donde se implementan los paquetes estándar debe aprovisionarse con una clase de almacenamiento predeterminada. Específicamente, los paquetes de Prometheus y Grafana requieren una clase de almacenamiento predeterminada. Si aprovisionó un clúster de TKG sin especificar la clase de almacenamiento predeterminada, puede aplicar revisiones a la clase de almacenamiento existente y agregar la anotación necesaria para especificarla como predeterminada. Consulte Aplicar revisiones a una clase de almacenamiento.

El límite de almacenamiento del espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG donde se van a instalar los paquetes de Tanzu debe ser mayor que el tamaño total de las notificaciones de volumen persistente. Para obtener más información sobre los requisitos de almacenamiento de espacio de nombres de vSphere, consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

**Tabla 11-1. Requisitos de almacenamiento persistente para paquetes estándar**

| Componente | Extensión TKG | Tamaño de almacenamiento predeterminado |
| --- | --- | --- |
| Grafana | Grafana | 8 Gi |
| Servidor Prometheus | Prometheus | 8 Gi |
| Alertmanager | Prometheus | 8 Gi |
| Harbor | Registro de Harbor | Varía según la PVC |

Para ajustar el límite de almacenamiento del espacio de nombres de vSphere donde se aprovisiona el TKG clúster:

1 Con vSphere Client, inicie sesión en la instancia de vCenter Server en la que está habilitado vSphere IaaS control plane.

2 Seleccione el espacio de nombres de vSphere donde se aprovisiona el clúster de Tanzu Kubernetes de destino.

3 Seleccione **Configurar > Límites de recursos**.

4 Haga clic en **Editar**.

5 Ajuste el límite de **Almacenamiento** para que sea mayor que el tamaño total de las notificaciones de volumen persistente que se requieren para las extensiones Prometheus y Grafana.

# Crear el repositorio de paquetes

Siga estas instrucciones para configurar el repositorio de paquetes estándar en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

## Requisitos

Cumpla los siguientes requisitos antes de crear el repositorio de paquetes.

- Requisitos generales
- Instalar el Herramientas de la CLI de Kubernetes para vSphere
- Instalar la instancia de CLI de Tanzu para usarla con clústeres de Servicio TKG
- Aprovisione un clúster de TKG mediante un TKr para vSphere 8.x. Consulte las notas de la versión de Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl y TKR.

## Instalar imgpkg de Carvel

La herramienta imgpkg de Carvel (https://carvel.dev/imgpkg/) le permite examinar las versiones disponibles del repositorio de paquetes estándar. El repositorio es público, por lo que no es necesario iniciar sesión.

Complete estos pasos para instalar imgpkg de Carvel.

1 Instale imgpkg mediante el siguiente comando.

```
wget -O- https://carvel.dev/install.sh > install.sh
sudo bash install.sh
```

2 Compruebe la instalación.

```
imgpkg version
```

Resultado de ejemplo:

```
imgpkg version 0.42.1
```

3 Para enumerar las versiones del repositorio, ejecute el siguiente comando:

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/repo
```

Este comando muestra las versiones disponibles del repositorio de paquetes estándar.

```
Tags

Name
...
v2024.4.12
v2024.4.19
v2024.5.14
v2024.5.16

39 tags

Succeeded
```

## Crear el repositorio de paquetes

Inicie sesión en el clúster de TKG y cree el repositorio de paquetes.

1 Inicie sesión en el clúster.

```
kubectl vsphere login --server=IP-or-FQDN --vsphere-username USER@vsphere.local --tanzu-
kubernetes-cluster-name CLUSTER --tanzu-kubernetes-cluster-namespace VSPHERE-NS
```

2 Cree el repositorio de paquetes.

```
tanzu package repository add standard-repo --url projects.registry.vmware.com/tkg/packages/
standard/repo:v2024.5.16 -n tkg-system
```

**Nota** Cambie la cadena de la versión del repositorio para que coincida con la versión del repositorio de destino.

3 Enumere los paquetes disponibles.

```
tanzu package available list -n tkg-system
```

**Nota** No todos los paquetes del repositorio son compatibles con clústeres de TKG. Consulte la lista oficial de paquetes compatibles: Paquetes estándar para clústeres de Servicio TKG.

4 Enumere las versiones disponibles para un paquete individual.

Administrador de certificados

```
tanzu package available get cert-manager.tanzu.vmware.com -n tkg-system
```

Contour

```
tanzu package available get contour.tanzu.vmware.com -n tkg-system
```

DNS externo

```
tanzu package available get external-dns.tanzu.vmware.com -n tkg-system
```

Fluent Bit

```
tanzu package available get fluent-bit.tanzu.vmware.com -n tkg-system
```

Grafana

```
tanzu package available get grafana.tanzu.vmware.com -n tkg-system
```

Prometheus

```
tanzu package available get prometheus.tanzu.vmware.com -n tkg-system
```

# Instalar administrador de certificados

Siga estas instrucciones para instalar el administrador de certificados en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

## Acerca del administrador de certificados

El administrador de certificados proporciona administración de certificados para clústeres de Servicio TKG. El administrador de certificados es un requisito previo para la mayoría de paquetes estándares, incluidos Contour, ExternalDNS, Prometheus y Harbor.

## Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales.\
- Crear el repositorio de paquetes

## Instalar administrador de certificados

Complete estos pasos para instalar el administrador de certificados.

1 Indique las versiones disponibles del administrador de certificados.

```
tanzu package available get cert-manager.tanzu.vmware.com -n tkg-system
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Cree el espacio de nombres del administrador de certificados.

```
kubectl create ns cert-manager
```

3 Instale el administrador de certificados.

Ajuste la versión de destino para cumplir con sus requisitos.

```
tanzu package install cert-manager -p cert-manager.tanzu.vmware.com -n cert-manager -v
1.12.2+vmware.2-tkg.2
```

4 Compruebe la instalación del administrador de certificados.

```
tanzu package installed list -n cert-manager
```

```
tanzu package installed get -n cert-manager cert-manager
```

5 Compruebe el espacio de nombres del administrador de certificados para ver los recursos creados por la instalación del paquete.

```
kubectl -n cert-manager get all
```

```
NAME                                          READY   STATUS    RESTARTS   AGE
pod/cert-manager-b5675b75f-flkjp              1/1     Running   0          6m14s
pod/cert-manager-cainjector-f8dc756cf-f7xsv   1/1     Running   0          6m14s
pod/cert-manager-webhook-6c888c8ddd-5xlnb     1/1     Running   0          6m14s

NAME                           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/cert-manager           ClusterIP   10.97.254.59     <none>        9402/TCP   6m14s
service/cert-manager-webhook   ClusterIP   10.105.225.156   <none>        443/TCP    6m14s

NAME                                      READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/cert-manager              1/1     1            1           6m14s
deployment.apps/cert-manager-cainjector   1/1     1            1           6m14s
deployment.apps/cert-manager-webhook      1/1     1            1           6m14s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/cert-manager-b5675b75f              1         1         1       6m14s
replicaset.apps/cert-manager-cainjector-f8dc756cf   1         1         1       6m14s
replicaset.apps/cert-manager-webhook-6c888c8ddd     1         1         1       6m14s
```

## Solucionar problemas

Utilice los siguientes comandos para buscar mensajes de error.

```
kubectl get pkgi -A
```

```
kubectl describe pkgi -n cert-manager cert-manage
```

# Instalar Contour con Envoy

Siga estas instrucciones para instalar Contour con Envoy en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

## Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales
- Referencia del paquete de Contour
- Crear el repositorio de paquetes
- Instalar administrador de certificados

## Crear valores de datos de Contour

Prepárese para instalar Contour creando el archivo de valores de datos.

1 Indique las versiones del paquete de Contour disponibles.

```
tanzu package available get contour.tanzu.vmware.com -n tkg-system
```

O bien, mediante el uso de kubectl:

```
kubectl -n tkg-system get packages | grep contour
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo `contour-default-values.yaml`.

```
tanzu package available get contour.tanzu.vmware.com/1.28.2+vmware.1-tkg.1 --default-values-file-output contour-data-values.yaml
```

Donde:

- *1.28.2+vmware.1-tkg.1* es la versión del paquete de destino
- *contour-data-values.yaml* es el nombre y la ruta del archivo de valores de datos que se va a generar

3 Edite el archivo `contour-data-values.yaml`.

Configure el servicio de Envoy en `LoadBalancer` para permitir que el tráfico de afuera del clúster acceda a un servicio de Kubernetes. Consulte el siguiente ejemplo para obtener instrucciones.

```
vi contour-data-values.yaml
```

```
---
infrastructure_provider: vsphere
namespace: tanzu-system-ingress
contour:
 configFileContents: {}
 useProxyProtocol: false
 replicas: 2
 pspNames: "vmware-system-restricted"
 logLevel: info
envoy:
 service:
   type: LoadBalancer
   annotations: {}
   externalTrafficPolicy: Cluster
   disableWait: false
 hostPorts:
   enable: true
   http: 80
   https: 443
```

```
 hostNetwork: false
 terminationGracePeriodSeconds: 300
 logLevel: info
certificates:
 duration: 8760h
 renewBefore: 360h
```

## Instalar Contour

Complete estos pasos para instalar la entrada de Contour con Envoy.

1 Cree un espacio de nombres exclusivo para el paquete de Contour.

```
kubectl create ns tanzu-system-ingress
```

2 Instale Contour.

Ajuste la versión para cumplir con sus requisitos.

```
tanzu package install contour -p contour.tanzu.vmware.com -v 1.28.2+vmware.1-tkg.1 --
values-file contour-data-values.yaml -n tanzu-system-ingress
```

3 Compruebe la instalación de Contour.

```
tanzu package installed list -n tanzu-system-ingress
```

4 Compruebe los objetos de Contour y Envoy.

```
kubectl -n tanzu-system-ingress get all
```

```
NAME                           READY   STATUS    RESTARTS   AGE
pod/contour-777bdddc69-fqnsp   1/1     Running   0          102s
pod/contour-777bdddc69-gs5xv   1/1     Running   0          102s
pod/envoy-d4jtt                2/2     Running   0          102s
pod/envoy-g5h72                2/2     Running   0          102s
pod/envoy-pjpzc                2/2     Running   0          102s

NAME              TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                       AGE
service/contour   ClusterIP      10.105.242.46   <none>
8001/TCP                      102s
service/envoy     LoadBalancer   10.103.245.57   10.197.154.69   80:32642/
TCP,443:30297/TCP   102s

NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/envoy   3         3         3       3            3
<none>          102s

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
```

```
deployment.apps/contour   2/2     2            2           102s

NAME                                 DESIRED   CURRENT   READY   AGE
replicaset.apps/contour-777bdddc69   2         2         2       102s
```

En este ejemplo, el servicio Envoy tiene una dirección IP externa 10.197.154.69. Esta dirección IP está separada del rango de CIDR especificado para **Red de carga de trabajo > Ingreso**. Se creará una nueva instancia de equilibrador de carga para esta dirección IP. Los miembros del grupo de servidores de este equilibrador de carga son los pods Envoy. Dado que los pods Envoy asumen las direcciones IP de los nodos de trabajo en los que se ejecutan, puede ver estas direcciones IP consultando los nodos del clúster (`kubectl get nodes -o wide`).

### Solucionar problemas

Consulte el siguiente tema según sea necesario.

- Referencia del paquete de Contour.

## Instalar ExternalDNS

Siga estas instrucciones para instalar ExternalDNS en un clúster Servicio TKG que ejecute TKr para vSphere 8.x.

### Acerca de ExternalDNS

ExternalDNS permite que los registros de DNS se creen automáticamente para los servicios de Kubernetes con un componente de entrada, como Contour con Envoy. El paquete de ExternalDNS se valida con los siguientes proveedores DNS: AWS Route 53, DNS de Azure y servidores DNS que cumplen con RFC2136 (como BIND). Consulte también Referencia del paquete ExternalDNS.

### Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales.
- Crear el repositorio de paquetes.
- Instalar administrador de certificados.
- Instalar Contour con Envoy.

### Crear valores de datos de ExternalDNS

Prepárese para instalar ExternalDNS; para ello, cree el archivo de valores de datos de ExternalDNS.

1 Enumere las versiones del paquete de ExternalDNS disponibles en el repositorio.

```
tanzu package available get external-dns.tanzu.vmware.com -n tkg-system
```

O bien, use kubectl.

```
kubectl -n tkg-system get packages | grep external-dns
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo de valores de datos para el paquete ExternalDNS.

```
tanzu package available get external-dns.tanzu.vmware.com/0.13.6+vmware.1-tkg.1 --default-
values-file-output external-dns-data-values.yaml
```

Donde:

- *0.13.6+vmware.1-tkg.1* es la versión del paquete de destino
- *external-dns-data-values.yaml* es el nombre y la ruta del archivo de valores de datos que se va a generar

3 Personalice los valores de los datos según sea necesario para su entorno.

Los valores de datos varían en función del servidor DNS compatible al que se dirige. Consulte Referencia del paquete ExternalDNS para ver ejemplos.

4 Si fuera necesario, cree un mapa de configuración que defina el servidor DNS con el que tendrá una interfaz el paquete ExternalDNS.

Consulte Referencia del paquete ExternalDNS para ver un ejemplo.

## Instalar ExternalDNS

Complete estos pasos para instalar el paquete ExternalDNS en un clúster de TKG.

1 Cree un espacio de nombres para ExternalDNS.

```
kubectl create ns tanzu-system-service-discovery
```

2 Instale el paquete de ExternalDNS mediante la CLI de Tanzu.

```
tanzu package install external-dns -p external-dns.tanzu.vmware.com -n tanzu-system-
service-discovery -v 0.11.0+vmware.1-tkg.2 --values-file external-dns-data-values.yaml
```

3 Compruebe que el paquete esté instalado mediante la CLI de Tanzu.

```
tanzu package installed list -n tanzu-system-service-discovery

NAME          PACKAGE-NAME                   PACKAGE-VERSION        STATUS
external-dns  external-dns.tanzu.vmware.com  0.11.0+vmware.1-tkg.2  Reconcile succeeded
```

```
kubectl -n tanzu-system-service-discovery get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/external-dns-77d947745-tcjz9    1/1     Running   0          63s

NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
```

```
deployment.apps/external-dns    1/1     1            1           63s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/external-dns-77d947745   1         1         1       63s
```

### Referencia

Consulte los siguientes temas según sea necesario.

- Referencia del paquete ExternalDNS.
- Instale ExternalDNS para la detección de servicios.

## Instalar Fluent Bit

Siga estas instrucciones para instalar Fluent Bit en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

### Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales
- Crear el repositorio de paquetes
- Instalar administrador de certificados
- Un destino compatible para el reenvío de registros de Fluent Bit. Consulte Referencia del paquete de Fluent Bit.

### Crear valores de datos de Fluent Bit

Prepárese para instalar Fluent Bit; para ello, cree el archivo de valores de datos.

1 Indique las versiones del paquete de Contour disponibles.

```
tanzu package available get fluent-bit.tanzu.vmware.com -n tkg-system
```

O bien, mediante el uso de kubectl:

```
kubectl -n tkg-system get packages | grep fluent-bit
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo `fluent-bit-data-values.yaml`.

```
tanzu package available get fluent-bit.tanzu.vmware.com/2.1.6+vmware.1-tkg.2 --default-values-file-output fluent-bit-data-values.yaml
```

3 Edite el archivo `fluent-bit-data-values.yaml` y configure los valores.

Consulte Referencia del paquete de Fluent Bit para obtener una lista de todos los parámetros disponibles.

### Instalar Fluent Bit

Complete estos pasos para instalar el paquete de Fluent Bit.

1 Cree el espacio de nombres para Fluent Bit.

```
kubectl create ns tanzu-system-logging
```

2 Instale Fluent Bit.

```
tanzu package install fluent-bit -p fluent-bit.tanzu.vmware.com -v 2.1.6+vmware.1-tkg.2
--values-file fluent-bit-data-values.yaml -n tanzu-system-logging
```

3 Compruebe la instalación de Fluent Bit.

```
tanzu package installed list -n tanzu-system-logging
```

```
tanzu package installed get fluent-bit -n tanzu-system-logging
```

4 Compruebe los objetos de Fluent Bit.

```
kubectl -n tanzu-system-logging get all
```

## Instalar Prometheus con Alertmanager

Siga estas instrucciones para instalar Prometheus con Alertmanager en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

### Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales
- Crear el repositorio de paquetes.
- Instalar administrador de certificados.
- Instalar Contour con Envoy (necesario para acceder al panel de control de Prometheus).
- Referencia del paquete de Prometheus

### Crear valores de datos de Prometheus

Prepárese para instalar Prometheus; para ello, cree el archivo de valores de datos.

1 Obtenga la versión más reciente del paquete de Prometheus para el repositorio.

```
tanzu package available get prometheus.tanzu.vmware.com -n tkg-system
```

O bien, use kubectl.

```
kubectl -n tkg-system get packages | grep prometheus
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo `prometheus-data-values.yaml`.

```
tanzu package available get prometheus.tanzu.vmware.com/2.45.0+vmware.1-tkg.2 --default-values-file-output prometheus-data-values.yaml
```

Donde:

- *2.45.0+vmware.1-tkg.2* es la versión del paquete de destino
- *prometheus-data-values.yaml* es el nombre y la ruta del archivo de los valores de datos que se va a generar

3 Edite el archivo de `prometheus-data-values.yaml` y configure los siguientes valores, que son necesarios para acceder al panel de control de Prometheus. Consulte Referencia del paquete de Prometheus para obtener un archivo de valores de datos de ejemplo y una lista completa de parámetros de configuración.

| Parámetro | Descripción |
|---|---|
| `ingress.tlsCertificate.tls.crt` | Se genera un certificado TLS autofirmado para la entrada. Opcionalmente, puede reemplazar y proporcionar uno propio. |
| `ingress.tlsCertificate.tls.key` | Se genera una clave privada TLS autofirmada para la entrada. Opcionalmente, puede reemplazar y proporcionar uno propio. |
| `ingress.enabled` | Establezca el valor en `true` (el valor predeterminado es false). |
| `ingress.virtual_host_fqdn` | Establezca el valor en `prometheus.<your.domain>` (el valor predeterminado es `prometheus.system.tanzu`). |
| `alertmanager.pvc.storageClassName` | Introduzca el nombre de la directiva de almacenamiento de vSphere. |
| `prometheus.pvc.storageClassName` | Introduzca el nombre de la directiva de almacenamiento de vSphere. |

## Instalar Prometheus

Siga estos pasos para instalar el paquete de Prometheus.

1 Cree el espacio de nombres.

```
kubectl create ns tanzu-system-monitoring
```

2 Instale Prometheus.

```
tanzu package install prometheus -p prometheus.tanzu.vmware.com -v 2.45.0+vmware.1-tkg.2
--values-file prometheus-data-values.yaml -n tanzu-system-monitoring
```

3 Compruebe la instalación de Prometheus.

```
tanzu package installed list -n tanzu-system-monitoring
```

```
tanzu package installed get prometheus -n tanzu-system-monitoring
```

4 Compruebe los objetos de Prometheus y de Altermanager.

```
kubectl -n tanzu-system-monitoring get all
```

```
kubectl -n tanzu-system-monitoring get pvc

NAME                STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS   AGE
alertmanager        Bound    pvc-a53f7091-9823-4b70-a9b4-c3d7a1e27a4b   2Gi
RWO            k8s-policy     2m30s
prometheus-server   Bound    pvc-41745d1d-9401-41d7-b44d-ba430ecc5cda   20Gi
RWO            k8s-policy     2m30s
```

## Solucionar problemas de la instalación de Prometheus

Si la operación `tanzu package install prometheus` devuelve el error "No se pudo obtener la dirección de anuncio final: no se encontró ninguna dirección IP privada y no se proporcionó una IP explícita", aplique una superposición de paquete para volver a configurar el componente alertmanager.

1 Cree el archivo `overlay-alertmanager.yaml`.

```
---
#@ load("@ytt:overlay", "overlay")

#@overlay/match by=overlay.and_op(overlay.subset({"kind": "Deployment"}),
overlay.subset({"metadata": {"name": "alertmanager"}}))
---
spec:
  template:
    spec:
      containers:
        #@overlay/match by="name",expects="0+"
        - name: alertmanager
          args:
            - --cluster.listen-address=
```

2 Utilice Kubectl para crear un secreto a partir del archivo `overlay-alertmanager.yaml`.

```
kubectl create secret generic alertmanager-overlay -n tkg-system -o yaml --dry-run=client
--from-file=overlay-alertmanager.yaml | kubectl apply -f -
```

3 Utilice Kubectl para anotar el paquete de Prometheus con el secreto de superposición.

```
kubectl annotate PackageInstall prometheus -n tkg-system ext.packaging.carvel.dev/ytt-
paths-from-secret-name.1=alertmanager-overlay
```

4 Vuelva a ejecutar el comando de instalación.

```
tanzu package install prometheus -p prometheus.tanzu.vmware.com -v 2.37.0+vmware.3-tkg.1
--values-file prometheus-data-values.yaml -n tanzu-system-monitoring
```

## Acceder al panel de control de Prometheus

Una vez que Prometheus esté instalado, complete los siguientes pasos para acceder al panel de control de Prometheus.

1 Asegúrese de que la sección `ingress` del archivo `prometheus-data-values.yaml` se rellena con todos los campos obligatorios.

```
ingress:
  enabled: true
  virtual_host_fqdn: "prometheus.system.tanzu"
  prometheus_prefix: "/"
  alertmanager_prefix: "/alertmanager/"
  prometheusServicePort: 80
  alertmanagerServicePort: 80
  #! [Optional] The certificate for the ingress if you want to use your own TLS
certificate.
  #! We will issue the certificate by cert-manager when it's empty.
  tlsCertificate:
    #! [Required] the certificate
    tls.crt:
    #! [Required] the private key
    tls.key:
    #! [Optional] the CA certificate
    ca.crt:
```

2 Obtenga la dirección IP pública (externa) del equilibrador de carga de Contour con Envoy.

Consulte Instalar Contour con Envoy.

3 Cree un registro de DNS que asigne el FQDN de Prometheus que utilizó (el valor predeterminado es `prometheus.system.tanzu`) a la dirección IP del equilibrador de carga de Envoy.

4 Para acceder al panel de control de Prometheus, desplácese hasta el FQDN de Prometheus con un navegador.

# Instalar Grafana

Siga estas instrucciones para instalar Grafana en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

## Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales.
- Crear el repositorio de paquetes
- Instalar administrador de certificados
- Instalar Contour con Envoy
- Instalar Prometheus con Alertmanager
- Referencia del paquete de Grafana

## Crear valores de datos de Grafana

Prepárese para instalar Grafana; para ello, cree el archivo de valores de datos.

1 Obtenga la versión más reciente del paquete de Prometheus para el repositorio.

```
tanzu package available get grafana.tanzu.vmware.com -n tkg-system
```

O bien, use kubectl.

```
kubectl -n tkg-system get packages | grep grafana
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo `prometheus-data-values.yaml`.

```
tanzu package available get grafana.tanzu.vmware.com/10.0.1+vmware.1-tkg.2 --default-
values-file-output grafana-data-values.yaml
```

Donde:

- *10.0.1+vmware.1-tkg.2* es la versión del paquete de destino
- *grafana-data-values.yaml* es el nombre y la ruta del archivo de valores de datos que se va a generar

3 Edite el archivo `gafana-data-values.yaml` y actualice los valores.

Agregue `ingress.pvc: storageClassName` y su valor, que es el nombre de la clase de almacenamiento de vSphere a la que puede acceder el clúster de TKG.

Para evitar un error común, elimine el secreto del archivo de valores de datos y cree manualmente el secreto. Consulte Solucionar problemas de la instalación de Grafana.

Este es un `gafana-data-values.yaml` mínimo, con el campo de clase de almacenamiento agregado y el secreto eliminado. Consulte Referencia del paquete de Grafana para obtener un ejemplo adicional y una lista completa de parámetros.

```
grafana:
  deployment:
    replicas: 1
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    storage: 2Gi
  service:
    port: 80
    targetPort: 3000
    type: LoadBalancer
ingress:
  enabled: true
  prefix: /
  servicePort: 80
  virtual_host_fqdn: grafana.system.tanzu
  pvc:
    storageClassName: vSphere-storage-profile
namespace: grafana
```

## Instalar Grafana

Siga estos pasos para instalar el paquete de Grafana.

1 Cree el espacio de nombres para Grafana.

```
kubectl create ns tanzu-system-dashboards
```

2 Instale el paquete de Grafana.

```
tanzu package install grafana -p grafana.tanzu.vmware.com -v 10.0.1+vmware.1-tkg.2 --
values-file grafana-data-values.yaml -n tanzu-system-dashboards
```

3 Compruebe la instalación de Grafana.

```
tanzu package installed list -n tanzu-system-dashboards
```

```
tanzu package installed get grafana -n tanzu-system-dashboards
```

4 Compruebe los objetos de Grafana.

```
kubectl -n tanzu-system-dashboards get all
```

5 Compruebe la notificación de volumen persistente que mantiene Grafana.

```
kubectl -n tanzu-system-dashboards get pvc
```

### Solucionar problemas de la instalación de Grafana

Para evitar el error "Secreto no creado al instalar Grafana desde el archivo YAML predeterminado", elimine `grafana.secret.*` de *grafana-data-values.yaml* y cree manualmente el secreto de la siguiente manera. A continuación, vuelva a implementar el paquete de Grafana.

```
kubectl create secret generic grafana -n tanzu-system-dashboards  --from-literal=admin=admin
```

## Instalar Registro de Harbor

Siga estas instrucciones para instalar el registro del contenedor de Harbor en un clúster de Servicio TKG que ejecute TKr para vSphere 8.x.

### Requisitos previos

Debe cumplir los siguientes requisitos previos.

- Requisitos generales
- Crear el repositorio de paquetes
- Instalar administrador de certificados
- Instalar Contour con Envoy
- Referencia del paquete de Harbor

### Crear valores de datos de Harbor

Cree el archivo de valores de datos a fin de prepararse para instalar Harbor.

1 Obtenga la versión más reciente del paquete de Harbor para el repositorio.

```
tanzu package available get harbor.tanzu.vmware.com -n tkg-system
```

O bien, use kubectl.

```
kubectl -n tkg-system get packages | grep harbor
```

**Nota** Por lo general, debe utilizar la versión más reciente, a menos que los requisitos sean diferentes.

2 Genere el archivo `harbor-data-values.yaml`.

```
tanzu package available get harbor.tanzu.vmware.com/2.9.1+vmware.1-tkg.1 --default-values-file-output harbor-data-values.yaml
```

Donde:

- *2.9.1+vmware.1-tkg.1* es la versión del paquete de destino
- *harbor-data-values.yaml* es el nombre y la ruta del archivo de valores de datos que se va a generar

3 Edite el archivo `harbor-data-values.yaml` y actualice los valores para los siguientes parámetros.

Configure parámetros adicionales según sea necesario. Consulte Referencia del paquete de Harbor.

| Campo | Descripción |
|---|---|
| `hostname` | El FQDN para acceder a la consola administrativa de Harbor y al servicio de registro. Reemplace "yourdomain.com" por un nombre de host único. |
| `harborAdminPassword` | Cambie la contraseña a una contraseña segura y única (también se puede cambiar en la interfaz de usuario después de instalar). |
| `persistence.persistentVolumeClaim.database.storageClass:` | Introduzca el nombre de la directiva de almacenamiento de vSphere para el espacio de nombres de vSphere. |
| `persistence.persistentVolumeClaim.jobservice.storageClass:` | Introduzca el nombre de la directiva de almacenamiento de vSphere para el espacio de nombres de vSphere. |
| `persistence.persistentVolumeClaim.redis.storageClass:` | Introduzca el nombre de la directiva de almacenamiento de vSphere para el espacio de nombres de vSphere. |
| `persistence.persistentVolumeClaim.registry.storageClass:` | Introduzca el nombre de la directiva de almacenamiento de vSphere para el espacio de nombres de vSphere. |
| `persistence.persistentVolumeClaim.trivy.storageClass:` | Introduzca el nombre de la directiva de almacenamiento de vSphere para el espacio de nombres de vSphere. |
| `tlsCertificate.tlsSecretLabels:` | {"managed-by": "vmware-vRegistry"} |

## Instalar Harbor

Complete los siguientes pasos para instalar el Registro de Harbor.

1 Cree el espacio de nombres para Harbor.

```
kubectl create ns tanzu-system-registry
```

2 Instale Harbor.

```
tanzu package install harbor --package harbor.tanzu.vmware.com --version 2.9.1+vmware.1-tkg.1 --values-file harbor-data-values.yaml --namespace tanzu-system-registry
```

3 Compruebe la instalación de Harbor.

```
tanzu package installed get harbor --namespace tanzu-system-registry
```

## Configurar DNS para Harbor mediante un servicio Envoy de tipo LoadBalancer

Si se expone el servicio Contour con Envoy como requisito previo a través de LoadBalancer, obtenga la dirección IP externa del equilibrador de carga y cree registros de DNS para los FQDN de Harbor.

1 Obtenga la dirección `External-IP` para el servicio Envoy de tipo LoadBalancer.

```
kubectl get service envoy -n tanzu-system-ingress
```

Debería ver la dirección `External-IP` que se devuelve, por ejemplo:

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

Si lo prefiere, puede obtener la dirección `External-IP` mediante el siguiente comando.

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Para comprobar la instalación de la extensión Harbor, actualice el archivo `/etc/hosts` local con los FQDN de Harbor y Notary asignados a la dirección `External-IP` del equilibrador de carga; por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
#TKG Harbor with Envoy Load Balancer IP
10.195.141.17 core.harbor.domain
10.195.141.17 core.notary.harbor.domain
```

3 Para comprobar la instalación de la extensión Harbor, inicie sesión en Harbor.

4 Cree dos registros CNAME en un servidor DNS que asignen la dirección `External-IP` del servicio Envoy del equilibrador de carga al FQDN de Harbor y al FQDN de Notary.

5 Instale la extensión DNS externo.

## Configurar DNS para Harbor mediante un servicio Envoy de tipo NodePort

Si se expone el servicio Contour > Envoy como requisito previo a través de NodePort, obtenga la dirección IP de la máquina virtual de un nodo de trabajo y cree registros de DNS para los FQDN de Harbor.

**Nota** Para usar NodePort, debe haber especificado el valor de `port.https` correcto en el archivo `harbor-data-values.yaml`.

1 Cambie el contexto a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster.

```
kubectl config use-context VSPHERE-NAMESPACE
```

2 Enumere los nodos del clúster.

```
kubectl get virtualmachines
```

3 Seleccione uno de los nodos de trabajo y describa el nodo mediante el siguiente comando.

```
kubectl describe virtualmachines tkg2-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 Busque la dirección IP de la máquina virtual; por ejemplo, `Vm Ip: 10.115.22.43`.

5 Para comprobar la instalación de la extensión Harbor, actualice el archivo `/etc/hosts` local con los FQDN de Harbor y Notary asignados a la dirección IP del nodo de trabajo; por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
#TKG Harbor with Envoy NodePort
10.115.22.43 core.harbor.domain
10.115.22.43 core.notary.harbor.domain
```

6 Para comprobar la instalación de la extensión Harbor, inicie sesión en Harbor.

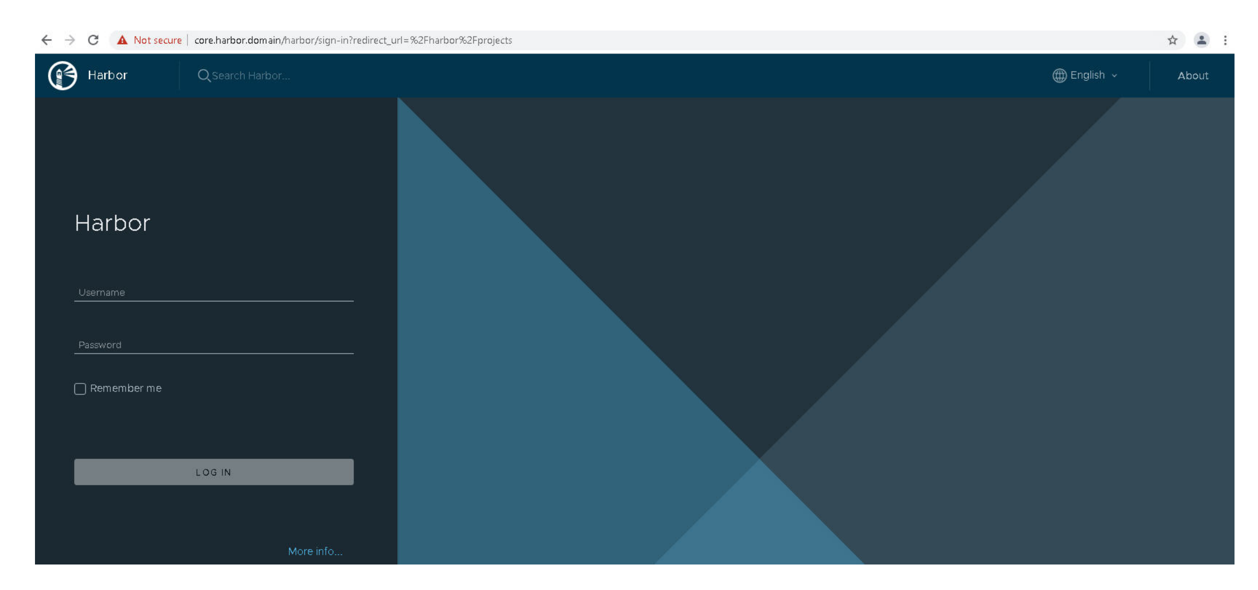
7 Cree dos registros CNAME en un servidor DNS que asignen la dirección IP del nodo de trabajo al FQDN de Harbor y al FQDN de Notary.

8 Instale la extensión DNS externo.

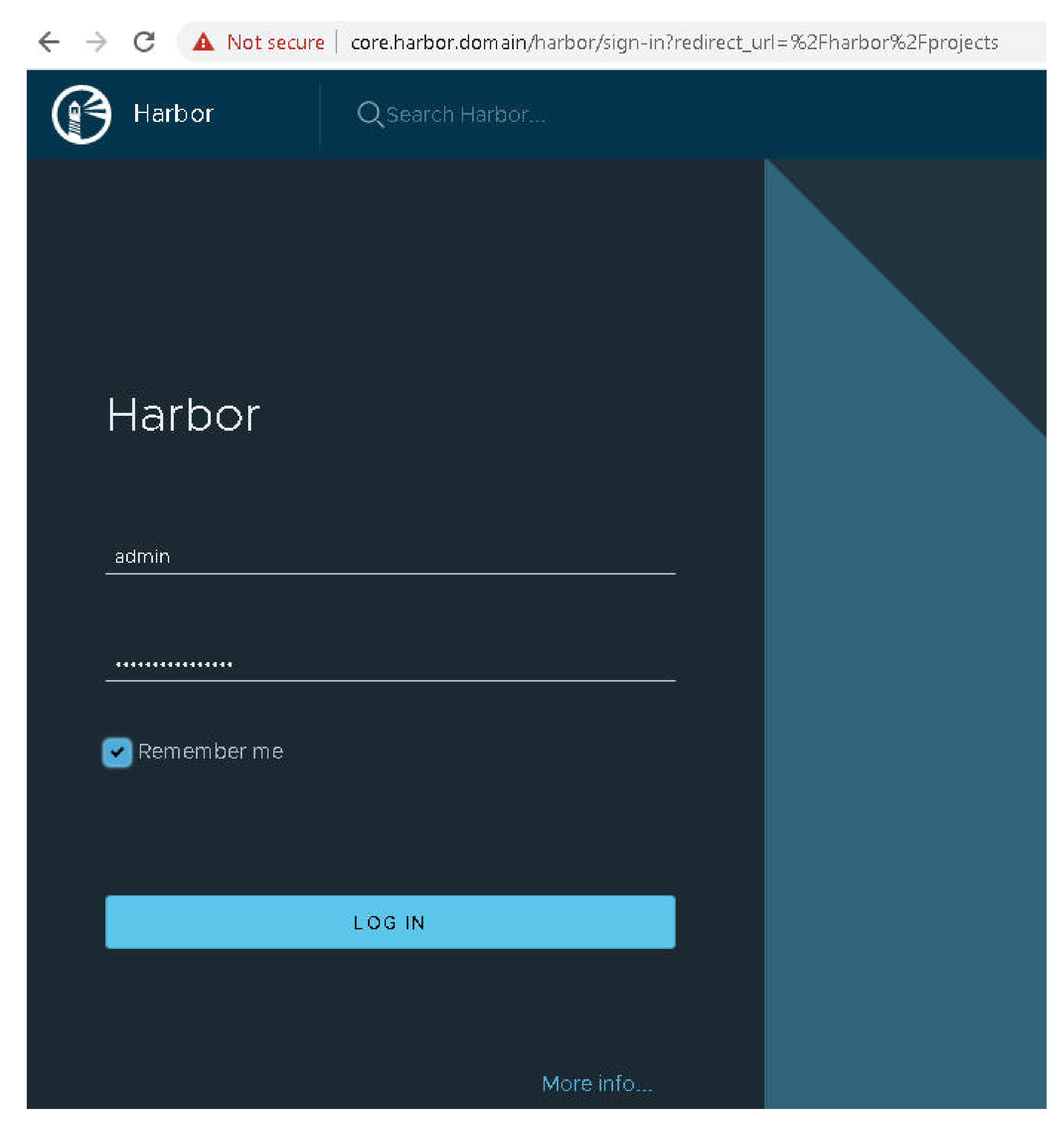
## Iniciar sesión en la interfaz web de Harbor

Una vez que Harbor esté instalado y configurado, inicie sesión y comience a utilizarlo.

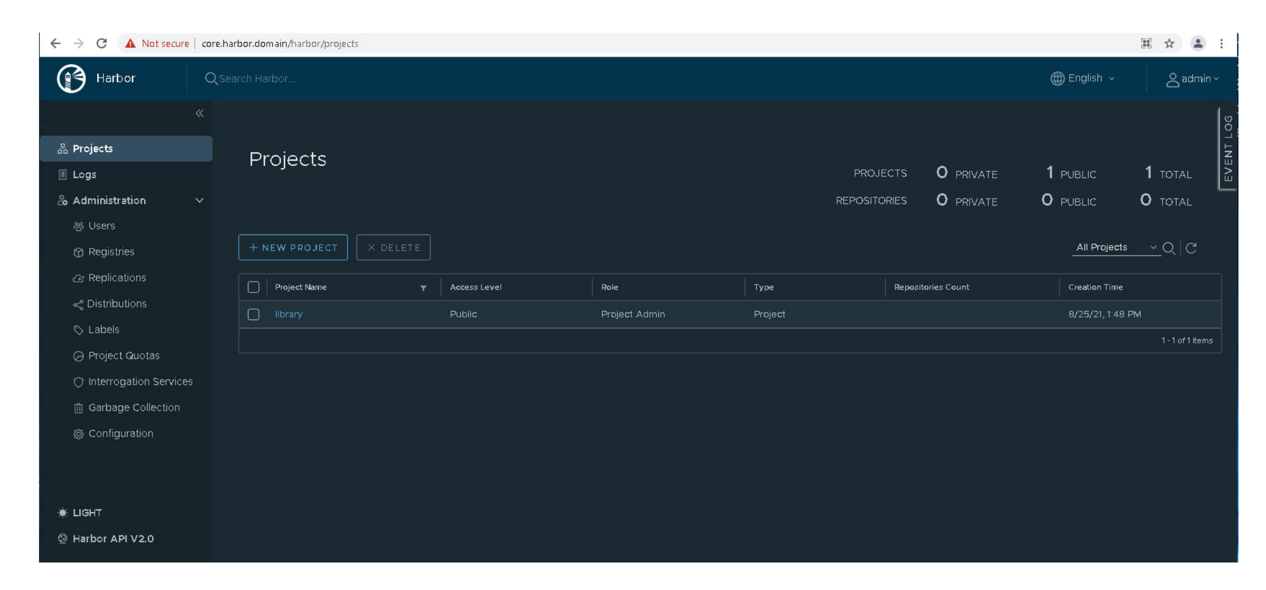
1 Acceda a la interfaz web del registro de Harbor en `https://core.harbor.domain` o al nombre de host que utilizó.




2 Inicie sesión en Harbor con el nombre de usuario **`admin`** y la contraseña generada que colocó en el archivo `harbor-data-values.yaml`.




3 Compruebe que puede acceder a la interfaz de usuario de Harbor.




4 Obtenga el certificado de CA de Harbor.

En la interfaz de Harbor, seleccione **Proyectos > biblioteca** o cree un **Nuevo proyecto**.

Haga clic en **Certificado del registro** y descargue el certificado de CA de Harbor (`ca.crt`).

5 Agregue el certificado de CA de Harbor al almacén de confianza del cliente de Docker para poder insertar y extraer imágenes de contenedor al registro de Harbor y desde él. Consulte Capítulo 14 Uso de registros privados con clústeres Servicio TKG.

6 Consulte la documentación de Harbor para obtener más información sobre el uso de Harbor.

# Referencia del paquete estándar

Esta sección proporciona información de referencia para los paquetes estándar que se pueden instalar en clústeres de Servicio TKG.

## Referencia del paquete de Contour

En este tema se proporciona información de referencia para el paquete Contour with Envoy.

### Acerca de Contour y Envory

Contour (https://projectcontour.io/) es un controlador de entrada de Kubernetes que incluye el proxy HTTP inverso Envoy. Contour con Envoy se utiliza habitualmente con otros paquetes, como ExternalDNS, Prometheus y Harbor.

Para instalar el paquete de Contour en un clúster de TKG, consulte los siguientes temas:

- Instalar Contour con Envoy
- #unique_173

## Componentes de Contour

El paquete de Contour incluye la controladora de entrada de Contour y el proxy HTTP inverso Envoy. Estos componentes se instalan como contenedores. Los contenedores se extraen del registro público especificado en el repositorio del paquete.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
|---|---|---|---|
| Envoy | DaemonSet | 3 | Proxy inverso de alto rendimiento |
| Contour | Implementación | 2 | Servidor de administración y configuración para Envoy |

## Valores de datos de Contour

A continuación se muestra un ejemplo `contour-data-values.yaml`.

La única personalización es que el servicio Envoy sea de tipo LoadBalancer (el valor predeterminado es NodePort). Esto significa que se podrá acceder al servicio Envoy desde fuera del clúster para la entrada.

```
infrastructure_provider: vsphere
namespace: tanzu-system-ingress
contour:
 configFileContents: {}
 useProxyProtocol: false
 replicas: 2
 pspNames: "vmware-system-restricted"
 logLevel: info
envoy:
 service:
   type: LoadBalancer
   annotations: {}
   nodePorts:
     http: null
     https: null
   externalTrafficPolicy: Cluster
   disableWait: false
 hostPorts:
   enable: true
   http: 80
   https: 443
 hostNetwork: false
 terminationGracePeriodSeconds: 300
 logLevel: info
 pspNames: null
certificates:
 duration: 8760h
 renewBefore: 360h
```

## Configuración de Contour

Los valores de configuración del paquete de Contour se establecen en `contour-data-values.yaml`. En la tabla se enumeran y se describen los parámetros disponibles.

### Tabla 11-2. Parámetros de configuración de entrada de Contour

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| infrastructure_provider | Proveedor de infraestructura. Valores admitidos: vsphere, aws, azure | string | Parámetro obligatorio |
| contour.namespace | Espacio de nombres en el que se implementará Contour | string | tanzu-system-ingress |
| contour.config.requestTimeout | Tiempo de espera de solicitud del cliente que se debe pasar a Envoy | time.Duration | 0s<br>(Consulte la sección a continuación para obtener más información) |
| contour.config.server.xdsServerType | Tipo de servidor XDS que se utilizará: valores compatibles: contour o envoy | string | Nulo |
| contour.config.tls.minimumProtocolVersion | Versión mínima de TLS que Contour negociará | string | 1,1 |
| contour.config.tls.fallbackCertificate.name | Nombre del secreto que contiene el certificado de reserva para las solicitudes que no coinciden con el SNI definido para un vhost | string | Nulo |
| contour.config.tls.fallbackCertificate.namespace | Espacio de nombres del secreto que contiene el certificado de reserva | string | Nulo |
| contour.config.tls.envoyClientCertificate.name | Nombre del secreto que se utilizará como certificado del cliente, clave privada para la conexión TLS al servicio back-end. | string | Nulo |
| contour.config.tls.envoyClientCertificate.namespace | Espacio de nombres del secreto que se utilizará como certificado del cliente, clave privada para la conexión TLS al servicio back-end. | string | Nulo |
| contour.config.leaderelection.configmapName | Nombre del configmap que se utilizará para contour leaderelection | string | leader-elect |
| contour.config.leaderelection.configmapNamespace | Espacio de nombres de contour leaderelection configmap | string | tanzu-system-ingress |
| contour.config.disablePermitInsecure | Deshabilita el campo ingressroute permitInsecure | booleano | false |

## Tabla 11-2. Parámetros de configuración de entrada de Contour (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| contour.config.accesslogFormat | Formato de registro de acceso | string | envoy |
| contour.config.jsonFields | Campos que se registrarán | array of strings | Documento del paquete de Envoy |
| contour.config.useProxyProtocol | https://projectcontour.io/guides/proxy-proto/ | booleano | false |
| contour.config.defaultHTTPVersions | Contour debería programar a Envoy para que sirva estas versiones HTTP | array of strings | "HTTP/1.1 HTTP2" |
| contour.config.timeouts.requestTimeout | El tiempo de espera de una solicitud completa | time.Duration | Nulo (el tiempo de espera está deshabilitado) |
| contour.config.timeouts.connectionIdleTimeout | El tiempo de espera antes de finalizar una conexión inactiva | time.Duration | 60s |
| contour.config.timeouts.streamIdleTimeout | Tiempo de espera antes de finalizar una solicitud o un flujo sin actividad | time.Duration | 5m |
| contour.config.timeouts.maxConnectionDuration | Tiempo de espera antes de finalizar una conexión, independientemente de si hay actividad o no | time.Duration | Nulo (el tiempo de espera está deshabilitado) |
| contour.config.timeouts.ConnectionShutdownGracePeriod | Tiempo que se debe esperar entre el envío de un GOAWAY inicial y final | time.Duration | 5s |
| contour.config.cluster.dnsLookupFamily | dns-lookup-family que se utilizará para las solicitudes de subida a servicios de tipo externalName desde una ruta HTTPProxy | string | Nulo (valores admitidos: auto, v4, v6) |
| contour.config.debug | Activa la depuración de Contour | booleano | false |
| contour.config.ingressStatusAddress | Dirección que se establecerá en el estado de cada recurso de entrada. | string | Nulo |
| contour.certificate.duration | Duración del certificado de Contour | time.Duration | 8760h |
| contour.certificate.renewBefore | Duración antes de la renovación del certificado de Contour | time.Duration | 360h |
| contour.deployment.replicas | Número de réplicas de Contour | entero | 2 |

## Tabla 11-2. Parámetros de configuración de entrada de Contour (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| contour.image.repository | Ubicación del repositorio con la imagen de Contour. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg |
| contour.image.name | Nombre de la imagen de Contour | string | contour |
| contour.image.tag | Etiqueta de la imagen de Contour. Es posible que este valor tenga que actualizarse si va a actualizar la versión de Contour. | string | v1.11.0_vmware.1 |
| contour.image.pullPolicy | Directiva de extracción de imagen de Contour | string | IfNotPresent |
| envoy.image.repository | Ubicación del repositorio con la imagen de Envoy. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg |
| envoy.image.name | Nombre de la imagen de Envoy | string | envoy |
| envoy.image.tag | Etiqueta de la imagen de Envoy. Es posible que este valor tenga que actualizarse si va a actualizar la versión de Envoy. | string | `v1.17.3_vmware.1` |
| envoy.image.pullPolicy | Directiva de extracción de la imagen de Envoy | string | IfNotPresent |
| envoy.hostPort.enable | Etiqueta para exponer los puertos de Envoy en el host | booleano | true |
| envoy.hostPort.http | Puerto de host HTTP de Envoy | entero | 80 |
| envoy.hostPort.https | Puerto de host HTTPS de Envoy | entero | 443 |

Tabla 11-2. Parámetros de configuración de entrada de Contour (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| envoy.service.type | Tipo de servicio para exponer Envoy. Valores admitidos: ClusterIP, NodePort y LoadBalancer | string | Parámetro obligatorio para vSphere: NodePort o LoadBalancer, AWS: LoadBalancer, Azure: LoadBalancer |
| envoy.service.annotations | Anotaciones en el servicio Envoy | Mapa (valores de clave) | Mapa vacío |
| envoy.service.externalTrafficPolicy | Directiva de tráfico externo del servicio Envoy. Valores admitidos: Local, Clúster | string | Clúster |
| envoy.service.nodePort.http | NodePort deseado para el servicio de tipo NodePort utilizado para las solicitudes http | entero | Nulo: Kubernetes asigna un puerto de nodo dinámico |
| envoy.service.nodePort.https | NodePort deseado para el servicio de tipo NodePort utilizado para las solicitudes HTTPS | entero | Nulo: Kubernetes asigna un puerto de nodo dinámico |
| envoy.deployment.hostNetwork | Ejecuta envoy en hostNetwork | booleano | false |
| envoy.service.aws.LBType | Tipo AWS LB que se utilizará para exponer el servicio Envoy. Valores admitidos: clásico, nlb | string | clásico |
| envoy.loglevel | Nivel de registro que se utilizará para Envoy | string | info |

## Tiempo de espera de ruta para las descargas de archivo

El parámetro `contour.config.requestTimeout` define la duración del tiempo de espera de la ruta de Contour. El valor predeterminado es `0s`. Si utiliza Contour para la transferencia de archivos, es posible que deba ajustar este valor.

Según la documentación de Contour , un valor de tiempo de espera de `0s` indica a Contour que utilice el tiempo de espera de Envoy. Según la documentación de Envoy, Envoy tiene un tiempo de espera de 15 segundos de forma predeterminada. Además, Envoy espera que toda la operación de solicitud-respuesta se complete dentro del intervalo de tiempo de espera.

Esto significa que con la configuración predeterminada de tiempo de espera de Contour de `0s`, la transferencia de archivos debe completarse en 15 segundos. Para las transferencias de archivos grandes, es posible que este tiempo no sea suficiente. Para deshabilitar el tiempo de espera predeterminado de Envoy, establezca el valor de `contour.config.requestTimeout` en `0`.

# Referencia del paquete ExternalDNS

Este tema proporciona información de referencia para el paquete ExternalDNS.

## Acerca de ExternalDNS

ExternalDNS sincroniza las entradas y los servicios de Kubernetes con los proveedores de DNS.

Consulte los siguientes temas para instalar ExternalDNS en un clúster de TKG.

- TKr para vSphere 8x: Instalar ExternalDNS
- TKr para vSphere 7.x: Instalar ExternalDNS

## Componentes ExternalDNS

El paquete ExternalDNS instala el contenedor que figura en la tabla. El paquete extrae el contenedor del registro público especificado en el repositorio de paquetes.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
| --- | --- | --- | --- |
| ExternalDNS | DaemonSet | 6 | Exponer servicios de Kubernetes para la búsqueda de DNS |

## Valores de datos de ExternalDNS

El archivo de valores de datos de ExternalDNS se utiliza para interconectarse al componente ExternalDNS con un proveedor de DNS compatible. El paquete de ExternalDNS se valida con los siguientes proveedores DNS: AWS (ruta 53), DNS de Azure y servidores DNS conformes a RFC2136 (como BIND).

El siguiente ejemplo se puede utilizar para un proveedor de DNS conforme a RFC2136 (como BIND).

```
---
# Namespace in which to deploy ExternalDNS pods
namespace: tanzu-system-service-discovery
# Deployment-related configuration
deployment:
args:
   - --registry=txt
   - --txt-owner-id=k8s
   - --txt-prefix=external-dns- #! Disambiguates TXT records from CNAME records
   - --provider=rfc2136
   - --rfc2136-host=IP-ADDRESS #! Replace with IP of RFC2136-compatible DNS server, such as 
192.168.0.1
   - --rfc2136-port=53
   - --rfc2136-zone=DNS-ZONE #! Replace with zone where services are deployed, such as my-
zone.example.org
   - --rfc2136-tsig-secret=TSIG-SECRET #! Replace with TSIG key secret authorized to update 
DNS server
   - --rfc2136-tsig-secret-alg=hmac-sha256
   - --rfc2136-tsig-keyname=TSIG-KEY-NAME #! Replace with TSIG key name, such as externaldns-
key
   - --rfc2136-tsig-axfr
```

```
    - --source=service
    - --source=ingress
    - --source=contour-httpproxy #! Enables Contour HTTPProxy object support
    - --domain-filter=DOMAIN #! Zone where services are deployed, such as my-zone.example.org
```

El siguiente ejemplo se puede utilizar para el proveedor de DNS de AWS (ruta 53).

```
---
    namespace: service-discovery
    dns:
      pspNames: "vmware-system-restricted"
      deployment:
        args:
        - --source=service
        - --source=ingress
        - --source=contour-httpproxy #! read Contour HTTPProxy resources
        - --domain-filter=my-zone.example.org #! zone where services are deployed
        - --provider=aws
        - --policy=upsert-only #! prevent deleting any records, omit to enable full
synchronization
        - --aws-zone-type=public #! only look at public hosted zones (public, private, no
value for both)
        - --aws-prefer-cname
        - --registry=txt
        - --txt-owner-id=HOSTED_ZONE_ID #! Route53 hosted zone identifier for my-
zone.example.org
        - --txt-prefix=txt #! disambiguates TXT records from CNAME records
        env:
          - name: AWS_ACCESS_KEY_ID
            valueFrom:
              secretKeyRef:
                name: route53-credentials #! Kubernetes secret for route53 credentials
                key: aws_access_key_id
          - name: AWS_SECRET_ACCESS_KEY
            valueFrom:
              secretKeyRef:
                name: route53-credentials #! Kubernetes secret for route53 credentials
                key: aws_secret_access_key
```

El siguiente ejemplo se puede utilizar para un proveedor de DNS de Azure.

```
---
    namespace: service-discovery
    dns:
      pspNames: "vmware-system-restricted"
      deployment:
        args:
        - --provider=azure
        - --source=service
        - --source=ingress
        - --source=contour-httpproxy #! read Contour HTTPProxy resources
        - --domain-filter=my-zone.example.org #! zone where services are deployed
        - --azure-resource-group=my-resource-group #! Azure resource group
        volumeMounts:
        - name: azure-config-file
```

```
          mountPath: /etc/kubernetes
          readOnly: true
        #@overlay/replace
        volumes:
        - name: azure-config-file
          secret:
            secretName: azure-config-file
```

## Configuración de ExternalDNS

En la tabla se enumeran y describen los parámetros de configuración disponibles para ExternalDNS. Consulte el sitio https://github.com/kubernetes-sigs/external-dns#running-externaldns para obtener más instrucciones.

### Tabla 11-3. Configuración de paquetes de DNS externos

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| externalDns.namespace | Espacio de nombres en el que se implementará el DNS externo | string | tanzu-system-service-discovery |
| externalDns.image.repository | Repositorio que contiene la imagen de DNS externo | string | projects.registry.vmware.com/tkg |
| externalDns.image.name | Nombre de external-dns | string | external-dns |
| externalDns.image.tag | Etiqueta de la imagen de DNS externo | string | v0.7.4_vmware.1 |
| externalDns.image.pullPolicy | Directiva de extracción de la imagen de DNS externo | string | IfNotPresent |
| externalDns.deployment.annotations | Anotaciones en la implementación de external-dns | map<string,string> | {} |
| externalDns.deployment.args | Argumentos transmitidos a través de la línea de comandos a external-dns | list<string> | `[]` (parámetro obligatorio) |
| externalDns.deployment.env | Variables de entorno que se transferirán a external-dns | list<string> | [] |
| externalDns.deployment.securityContext | Contexto de seguridad del contenedor de DNS externo | SecurityContext | {} |
| externalDns.deployment.volumeMounts | Montajes de volumen del contenedor de external-dns | list<VolumeMount> | [] |
| externalDns.deployment.volumes | Volúmenes del pod de DNS externo | list<Volume> | [] |

## Configmap de ejemplo

En el siguiente ejemplo, configmap define una configuración de Kerberos con la que ExternalDNS puede establecer una interfaz. Las entradas personalizadas incluyen el nombre de dominio/realm y las direcciones kdc/admin_server.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: krb.conf
  namespace: tanzu-system-service-discovery
data:
  krb5.conf: |
    [logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

    [libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = /etc/pki/tls/certs/ca-bundle.crt
    default_ccache_name = KEYRING:persistent:%{uid}

    default_realm = CORP.ACME

    [realms]
    CORP.ACME = {
      kdc = controlcenter.corp.acme
      admin_server = controlcenter.corp.acme
    }

    [domain_realm]
    corp.acme = CORP.ACME
    .corp.acme = CORP.ACME
```

# Referencia del paquete de Fluent Bit

Este tema proporciona información de referencia para el paquete de Fluent Bit.

## Acerca de Fluent Bit

Fluent Bit (https://fluentbit.io/) es un reenviador y procesador de registros rápido y ligero que le permite recopilar registros y datos de aplicaciones de diferentes orígenes, así como unificarlos y enviarlos a varios destinos.

Puede utilizar Fluent Bit como reenviador de registros para registros de un clúster de TKG. Debe tener un servidor de administración de registros implementado para almacenar y analizar registros. Los servidores de registro compatibles incluyen Syslog, HTTP, Elastic Search, Kafka y Splunk.

Para instalar el paquete de Fluent Bit en un clúster de TKG, consulte los siguientes temas:

- TKr de vSphere 8.x: Instalar Fluent Bit
- TKr de vSphere 7.x: #unique_175

## Componentes de Fluent Bit

El paquete Fluent Bit instala en el clúster el contenedor enumerado en la tabla. El paquete extrae el contenedor del registro público especificado en el repositorio de paquetes.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
| --- | --- | --- | --- |
| Fluent Bit | DaemonSet | 6 | Recopilador de registros, agregador, reenviador |

## Valores de datos de Fluent Bit

El siguiente ejemplo `fluent-bit-data-values.yaml` se puede utilizar para servidores syslog.

```
---
namespace: fluentbit-logging
tkg:
  instance_name: "<TKG_INSTANCE_NAME>"
  cluster_name: "<CLUSTER_NAME>"
fluentbit:
  output_plugin: "syslog"
  syslog:
    host: "<SYSLOG_HOST>"
    port: "<SYSLOG_PORT>"
    mode: "<SYSLOG_MODE>"
    format: "<SYSLOG_FORMAT>"
```

El siguiente ejemplo `fluent-bit-data-values.yaml` se puede utilizar para los endpoints de HTTP.

```
---
namespace: fluentbit-logging
tkg:
  instance_name: "<TKG_INSTANCE_NAME>"
  cluster_name: "<CLUSTER_NAME>"
fluentbit:
  output_plugin: "http"
  http:
    host: "<HTTP_HOST>"
    port: "<HTTP_PORT>"
    uri: "<URI>"
    header_key_value: "<HEADER_KEY_VALUE>"
    format: "json"
```

El siguiente ejemplo `fluent-bit-data-values.yaml` se puede utilizar para Elastic Search.

```
---
namespace: fluentbit-logging
tkg:
  instance_name: "<TKG_INSTANCE_NAME>"
```

```
  cluster_name: "<CLUSTER_NAME>"
fluentbit:
  output_plugin: "elasticsearch"
  elasticsearch:
    host: "<ELASTIC_SEARCH_HOST>"
    port: "<ELASTIC_SEARCH_PORT>"
```

El siguiente ejemplo `fluent-bit-data-values.yaml` se puede utilizar para Kafka.

```
---
namespace: fluentbit-logging
tkg:
  instance_name: "<TKG_INSTANCE_NAME>"
  cluster_name: "<CLUSTER_NAME>"
fluentbit:
  output_plugin: "kafka"
  kafka:
    broker_service_name: "<BROKER_SERVICE_NAME>"
    topic_name: "<TOPIC_NAME>"
```

El siguiente ejemplo `fluent-bit-data-values.yaml` se puede utilizar para Splunk.

```
---
namespace: fluentbit-logging
tkg:
  instance_name: "<TKG_INSTANCE_NAME>"
  cluster_name: "<CLUSTER_NAME>"
fluentbit:
  output_plugin: "splunk"
  splunk:
    host: "<SPLUNK_HOST>"
    port: "<SPLUNK_PORT>"
    token: "<SPLUNK_TOKEN>"
```

## Configuración de Fluent Bit

Los valores de configuración se establecen en `fluent-bit-data-values.yaml`. En la tabla se enumeran y se describen los parámetros disponibles.

### Tabla 11-4. Configuraciones del paquete de Fluent Bit

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| logging.namespace | Espacio de nombres en el que se implementará Fluent Bit | string | tanzu-system-logging |
| logging.service_account_name | Nombre de la cuenta del servicio Fluent Bit | string | fluent-bit |
| logging.cluster_role_name | Nombre de la función de clúster que otorga permisos para obtener, ver y enumerar Fluent Bit | string | fluent-bit-read |

## Tabla 11-4. Configuraciones del paquete de Fluent Bit (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| logging.image.name | Nombre de la imagen de Fluent Bit | string | fluent-bit |
| logging.image.tag | Etiqueta de la imagen de Fluent Bit. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v1.6.9_vmware.1 |
| logging.image.repository | Ubicación del repositorio con la imagen de Fluent Bit. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg |
| logging.image.pullPolicy | Directiva de extracción de imágenes de Fluent Bit | string | IfNotPresent |
| logging.update_strategy | Estrategia de actualización que se utilizará al actualizar DaemonSet | string | RollingUpdate |
| tkg.cluster_name | Nombre del clúster de Tanzu Kubernetes | string | Nulo (parámetro obligatorio) |
| tkg.instance_name | Nombre definido por el usuario de la instancia de TKG compartida por el clúster supervisor y todos los clústeres de Tanzu Kubernetes en una implementación. Puede utilizar cualquier nombre relacionado con la instalación. | string | Nulo (parámetro obligatorio)<br>**Nota** Este campo es obligatorio pero arbitrario. Es un nombre que aparece en los registros. |
| fluent_bit.log_level | Nivel de registro que se utilizará para Fluent Bit | string | info |
| fluent_bit.output_plugin | Establezca el back-end en el que Fluent Bit debería vaciar la información que recopila | string | Nulo (parámetro obligatorio) |
| fluent_bit.elasticsearch.host | Dirección IP o nombre de host de la instancia Elasticsearch de destino | string | Nulo (parámetro obligatorio cuando output_plugin es una búsqueda elástica) |
| fluent_bit.elasticsearch.port | Puerto TCP de la instancia Elasticsearch de destino | entero | Nulo (parámetro obligatorio cuando output_plugin es una búsqueda elástica) |

## Tabla 11-4. Configuraciones del paquete de Fluent Bit (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| fluent_bit.elasticsearch.buffer_size | Especifique el tamaño de búfer utilizado para leer la respuesta del servicio Elasticsearch. Se establece en ilimitado si es False | string | False |
| fluent_bit.elasticsearch.tls | Especifique la configuración predeterminada de TLS para Elasticsearch | string | Desactivado |
| fluent_bit.kafka.broker_service_name | Lista única o múltiple de Kafka Brokers; por ejemplo, 192.168.1.3:9092 | string | Nulo (parámetro obligatorio cuando output_plugin es kafka) |
| fluent_bit.kafka.topic_name | Entrada única o lista de temas separados por (,) que Fluent Bit usará para enviar mensajes a Kafka | string | Nulo (parámetro obligatorio cuando output_plugin es kafka) |
| fluent_bit.splunk.host | Dirección IP o nombre de host del servidor Splunk de destino | string | Nulo (parámetro obligatorio cuando output_plugin es splunk) |
| fluent_bit.splunk.port | Puerto TCP del servidor Splunk de destino | entero | Nulo (parámetro obligatorio cuando output_plugin es splunk) |
| fluent_bit.splunk.token | Especifica el token de autenticación de la interfaz del recopilador de eventos HTTP | string | Nulo (parámetro obligatorio cuando output_plugin es splunk) |
| fluent_bit.http.host | Dirección IP o nombre de host del servidor HTTP de destino | string | Nulo (parámetro obligatorio cuando output_plugin es http) |
| fluent_bit.http.port | Puerto TCP del servidor HTTP de destino | entero | Nulo (parámetro obligatorio cuando output_plugin es http) |
| fluent_bit.http.mode | Especificar un URI HTTP para el servidor web de destino | string | Nulo (parámetro obligatorio cuando output_plugin es http) |
| fluent_bit.http.header_key_value | Par clave/valor de encabezado HTTP. Se pueden establecer varios encabezados | string | Nulo (parámetro obligatorio cuando output_plugin es http) |
| fluent_bit.http.format | Especifica el formato de datos que se utilizará en el cuerpo de la solicitud HTTP | string | Nulo (parámetro obligatorio cuando output_plugin es http) |
| fluent_bit.syslog.host | Dominio o dirección IP del servidor Syslog remoto | string | Nulo (parámetro obligatorio cuando output_plugin es syslog) |

Tabla 11-4. Configuraciones del paquete de Fluent Bit (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| fluent_bit.syslog.port | Puerto TCP o UDP del servidor Syslog remoto | entero | Nulo (parámetro obligatorio cuando output_plugin es syslog) |
| fluent_bit.syslog.mode | Especifique el tipo de transporte de TCP, UDP y TLS | string | Nulo (parámetro obligatorio cuando output_plugin es syslog) |
| fluent_bit.syslog.format | Especifica el formato de datos que se utilizará en el cuerpo de la solicitud HTTP | string | Nulo (parámetro obligatorio cuando output_plugin es syslog) |
| host_path.volume_1 | Ruta de directorio desde el sistema de archivos del nodo host hacia el pod, para el volumen 1 | string | /var/log |
| host_path.volume_2 | Ruta de directorio desde el sistema de archivos del nodo host hacia el pod, para el volumen 2 | string | /var/lib/docker/containers |
| host_path.volume_3 | Ruta de directorio desde el sistema de archivos del nodo host hacia el pod, para el volumen 3 | string | /run/log |
| systemd.path | Ruta de acceso al directorio de diario Systemd | string | /var/log/journal |

# Referencia del paquete de Prometheus

En este tema se proporciona información de referencia para el paquete de Prometheus.

## Acerca de Prometheus y Altertmanager

Prometheus (https://prometheus.io/) es un sistema de supervisión de sistemas y servicios. Prometheus recopila métricas de los destinos configurados en intervalos determinados, evalúa las expresiones de reglas y muestra los resultados. Alertmanager se utiliza para activar alertas si se observa que alguna condición es verdadera.

Para instalar el paquete de Prometheus, haga lo siguiente:

-
-

## Componentes de Prometheus

El paquete Prometheus instala en un clúster de TKG los contenedores enumerados en la tabla. El paquete extrae los contenedores del registro público de VMware especificado en el repositorio de paquetes.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
|---|---|---|---|
| `prometheus-alertmanager` | Implementación | 1 | Controla las alertas enviadas por las aplicaciones cliente, como el servidor Prometheus. |
| `prometheus-cadvisor` | DaemonSet | 5 | Analiza y expone datos de rendimiento y uso de recursos de los contenedores que se están ejecutando. |
| `prometheus-kube-state-metrics` | Implementación | 1 | Supervisa el estado y la capacidad del nodo, la conformidad del conjunto de réplicas, el pod, el estado del trabajo y el trabajo cron, las solicitudes de recursos y los límites. |
| `prometheus-node-exporter` | DaemonSet | 5 | Exportador de métricas de hardware y SO expuestas por los kernels. |
| `prometheus-pushgateway` | Implementación | 1 | Servicio que le permite insertar métricas de los trabajos que no se pueden extraer. |
| `prometheus-server` | Implementación | 1 | Proporciona funcionalidades básicas, como la extracción, el procesamiento de reglas y las alertas. |

## Valores de datos de Prometheus

A continuación se muestra un archivo de ejemplo `prometheus-data-values.yaml`.

Tenga en cuenta lo siguiente:

- La entrada está habilitada (ingress: enabled: true).
- La entrada está configurada para las URL que terminan en /alertmanager/ (alertmanager_prefix:) y / (prometheus_prefix:).
- El FQDN de Prometheus es `prometheus.system.tanzu` (virtual_host_fqdn:).
- Proporcione su propio certificado personalizado en la sección de entrada (tls.crt, tls.key, ca.crt).
- La PVC para Alertmanager es 2GiB. Proporcione los `storageClassName` para la directiva de almacenamiento predeterminada.
- La PVC para Prometheus es de 20 GiB. Proporcione la `storageClassName` para la directiva de almacenamiento de vSphere.

```
namespace: prometheus-monitoring
alertmanager:
  config:
    alertmanager_yml: |
      global: {}
      receivers:
      - name: default-receiver
      templates:
      - '/etc/alertmanager/templates/*.tmpl'
      route:
```

```
        group_interval: 5m
        group_wait: 10s
        receiver: default-receiver
        repeat_interval: 3h
  deployment:
    replicas: 1
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    storage: 2Gi
    storageClassName: default
  service:
    port: 80
    targetPort: 9093
    type: ClusterIP
ingress:
  alertmanager_prefix: /alertmanager/
  alertmanagerServicePort: 80
  enabled: true
  prometheus_prefix: /
  prometheusServicePort: 80
  tlsCertificate:
    ca.crt: |
      -----BEGIN CERTIFICATE-----
      MIIFczCCA1ugAwIBAgIQTYJITQ3SZ4BBS9UzXfJIuTANBgkqhkiG9w0BAQsFADBM
      ...
      w0oGuTTBfxSMKs767N3G1q5tz0mwFpIqIQtXUSmaJ+9p7IkpWcThLnyYYo1IpWm/
      ZHtjzZMQVA==
      -----END CERTIFICATE-----
    tls.crt: |
      -----BEGIN CERTIFICATE-----
      MIIHxTCCBa2gAwIBAgITIgAAAAQnSpH7QfxTKAAAAAAABDANBgkqhkiG9w0BAQsF
      ...
      YYsIjp7/f+Pk1DjzWx8JIAbzItKLucDreAmmDXqk+DrBP9LYqtmjB0n7nSErgK8G
      sA3kGCJdOkI0kgF10gsinaouG2jVlwNOsw==
      -----END CERTIFICATE-----
    tls.key: |
      -----BEGIN PRIVATE KEY-----
      MIIJRAIBADANBgkqhkiG9w0BAQEFAASCCS4wggkqAgEAAoICAQDOGHT8I12KyQGS
      ...
      l1NzswracGQIzo03zk/X3Z6P2YOea4BkZ0Iwh34wOHJnTkfEeSx6y+oSFMcFRthT
      yfFCZUk/sVCc/C1a4VigczXftUGiRrTR
      -----END PRIVATE KEY-----
  virtual_host_fqdn: prometheus.system.tanzu
kube_state_metrics:
  deployment:
    replicas: 1
  service:
    port: 80
    targetPort: 8080
    telemetryPort: 81
    telemetryTargetPort: 8081
```

```
    type: ClusterIP
node_exporter:
  daemonset:
    hostNetwork: false
    updatestrategy: RollingUpdate
  service:
    port: 9100
    targetPort: 9100
    type: ClusterIP
prometheus:
  pspNames: "vmware-system-restricted"
  config:
    alerting_rules_yml: |
      {}
    alerts_yml: |
      {}
    prometheus_yml: |
      global:
        evaluation_interval: 1m
        scrape_interval: 1m
        scrape_timeout: 10s
      rule_files:
      - /etc/config/alerting_rules.yml
      - /etc/config/recording_rules.yml
      - /etc/config/alerts
      - /etc/config/rules
      scrape_configs:
      - job_name: 'prometheus'
        scrape_interval: 5s
        static_configs:
        - targets: ['localhost:9090']
      - job_name: 'kube-state-metrics'
        static_configs:
        - targets: ['prometheus-kube-state-metrics.prometheus.svc.cluster.local:8080']

      - job_name: 'node-exporter'
        static_configs:
        - targets: ['prometheus-node-exporter.prometheus.svc.cluster.local:9100']

      - job_name: 'kubernetes-pods'
        kubernetes_sd_configs:
        - role: pod
        relabel_configs:
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
          action: keep
          regex: true
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
          action: replace
          target_label: __metrics_path__
          regex: (.+)
        - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
          action: replace
          regex: ([^:]+)(?::\d+)?;(\d+)
          replacement: $1:$2
          target_label: __address__
```

```
      - action: labelmap
        regex: __meta_kubernetes_pod_label_(.+)
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
        target_label: kubernetes_namespace
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: kubernetes_pod_name
    - job_name: kubernetes-nodes-cadvisor
      kubernetes_sd_configs:
      - role: node
      relabel_configs:
      - action: labelmap
        regex: __meta_kubernetes_node_label_(.+)
      - replacement: kubernetes.default.svc:443
        target_label: __address__
      - regex: (.+)
        replacement: /api/v1/nodes/$1/proxy/metrics/cadvisor
        source_labels:
        - __meta_kubernetes_node_name
        target_label: __metrics_path__
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    - job_name: kubernetes-apiservers
      kubernetes_sd_configs:
      - role: endpoints
      relabel_configs:
      - action: keep
        regex: default;kubernetes;https
        source_labels:
        - __meta_kubernetes_namespace
        - __meta_kubernetes_service_name
        - __meta_kubernetes_endpoint_port_name
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
    alerting:
      alertmanagers:
      - scheme: http
        static_configs:
        - targets:
          - alertmanager.prometheus.svc:80
      - kubernetes_sd_configs:
          - role: pod
        relabel_configs:
        - source_labels: [__meta_kubernetes_namespace]
          regex: default
          action: keep
        - source_labels: [__meta_kubernetes_pod_label_app]
          regex: prometheus
```

```
            action: keep
          - source_labels: [__meta_kubernetes_pod_label_component]
            regex: alertmanager
            action: keep
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_probe]
            regex: .*
            action: keep
          - source_labels: [__meta_kubernetes_pod_container_port_number]
            regex:
            action: drop
    recording_rules_yml: |
      groups:
        - name: kube-apiserver.rules
          interval: 3m
          rules:
          - expr: |2
              (
                (
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[1d]))
                  -
                  (
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[1d]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[1d]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[1d]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[1d]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[1d]))
            labels:
              verb: read
            record: apiserver_request:burnrate1d
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[1h]))
                  -
                  (
```

```
                      (
                        sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[1h]))
                        or
                        vector(0)
                      )
                      +
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[1h]))
                      +
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[1h]))
                    )
                  )
                  +
                  # errors
                  sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[1h]))
                )
                /
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[1h]))
            labels:
              verb: read
            record: apiserver_request:burnrate1h
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[2h]))
                  -
                  (
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[2h]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[2h]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[2h]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[2h]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[2h]))
```

```
            labels:
              verb: read
            record: apiserver_request:burnrate2h
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[30m]))
                  -
                  (
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[30m]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[30m]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[30m]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[30m]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[30m]))
            labels:
              verb: read
            record: apiserver_request:burnrate30m
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[3d]))
                  -
                  (
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[3d]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[3d]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
```

```
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[3d]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[3d]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[3d]))
            labels:
              verb: read
            record: apiserver_request:burnrate3d
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[5m]))
                  -
                  (
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[5m]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[5m]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[5m]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[5m]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[5m]))
            labels:
              verb: read
            record: apiserver_request:burnrate5m
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[6h]))
                  -
                  (
```

```
                    (
                      sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[6h]))
                      or
                      vector(0)
                    )
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[6h]))
                    +
                    sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[6h]))
                  )
                )
                +
                # errors
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|
GET",code=~"5.."}[6h]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"LIST|GET"}
[6h]))
            labels:
              verb: read
            record: apiserver_request:burnrate6h
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[1d]))
                  -
                  sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[1d]))
                )
                +
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[1d]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[1d]))
            labels:
              verb: write
            record: apiserver_request:burnrate1d
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[1h]))
                  -
                  sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[1h]))
                )
```

```
                 +
                 sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[1h]))
               )
               /
               sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[1h]))
             labels:
               verb: write
             record: apiserver_request:burnrate1h
           - expr: |2
               (
                 (
                   # too slow
                   sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[2h]))
                   -
                   sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[2h]))
                 )
                 +
                 sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[2h]))
               )
               /
               sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[2h]))
             labels:
               verb: write
             record: apiserver_request:burnrate2h
           - expr: |2
               (
                 (
                   # too slow
                   sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[30m]))
                   -
                   sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[30m]))
                 )
                 +
                 sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[30m]))
               )
               /
               sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[30m]))
             labels:
               verb: write
             record: apiserver_request:burnrate30m
           - expr: |2
               (
                 (
                   # too slow
                   sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
```

```
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[3d]))
                  -
                  sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[3d]))
                )
                +
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[3d]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[3d]))
            labels:
              verb: write
            record: apiserver_request:burnrate3d
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[5m]))
                  -
                  sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[5m]))
                )
                +
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[5m]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[5m]))
            labels:
              verb: write
            record: apiserver_request:burnrate5m
          - expr: |2
              (
                (
                  # too slow
                  sum(rate(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[6h]))
                  -
                  sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE",le="1"}[6h]))
                )
                +
                sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE",code=~"5.."}[6h]))
              )
              /
              sum(rate(apiserver_request_total{job="kubernetes-apiservers",verb=~"POST|PUT|
PATCH|DELETE"}[6h]))
            labels:
              verb: write
            record: apiserver_request:burnrate6h
```

```
          - expr: |
              sum by (code,resource) (rate(apiserver_request_total{job="kubernetes-
apiservers",verb=~"LIST|GET"}[5m]))
            labels:
              verb: read
            record: code_resource:apiserver_request_total:rate5m
          - expr: |
              sum by (code,resource) (rate(apiserver_request_total{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[5m]))
            labels:
              verb: write
            record: code_resource:apiserver_request_total:rate5m
          - expr: |
              histogram_quantile(0.99, sum by
(le, resource) (rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET"}[5m]))) > 0
            labels:
              quantile: "0.99"
              verb: read
            record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile
          - expr: |
              histogram_quantile(0.99, sum by
(le, resource) (rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"POST|PUT|PATCH|DELETE"}[5m]))) > 0
            labels:
              quantile: "0.99"
              verb: write
            record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile
          - expr: |2
              sum(rate(apiserver_request_duration_seconds_sum{subresource!="log",verb!~"LIST|
WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT"}[5m])) without(instance, pod)
              /
              sum(rate(apiserver_request_duration_seconds_count{subresource!="log",verb!
~"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT"}[5m])) without(instance, pod)
            record: cluster:apiserver_request_duration_seconds:mean5m
          - expr: |
              histogram_quantile(0.99,
sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-apiservers",subresource!
="log",verb!~"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT"}[5m])) without(instance,
pod))
            labels:
              quantile: "0.99"
            record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile
          - expr: |

histogram_quantile(0.9, sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",subresource!="log",verb!~"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT"}
[5m])) without(instance, pod))
            labels:
              quantile: "0.9"
            record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile
          - expr: |

histogram_quantile(0.5, sum(rate(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",subresource!="log",verb!~"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT"}
```

```
[5m])) without(instance, pod))
            labels:
              quantile: "0.5"
            record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile
        - interval: 3m
          name: kube-apiserver-availability.rules
          rules:
          - expr: |2
              1 - (
                (
                  # write too slow
                  sum(increase(apiserver_request_duration_seconds_count{verb=~"POST|PUT|PATCH|
DELETE"}[30d]))
                  -
                  sum(increase(apiserver_request_duration_seconds_bucket{verb=~"POST|PUT|
PATCH|DELETE",le="1"}[30d]))
                ) +
                (
                  # read too slow
                  sum(increase(apiserver_request_duration_seconds_count{verb=~"LIST|GET"}
[30d]))
                  -
                  (
                    (
                      sum(increase(apiserver_request_duration_seconds_bucket{verb=~"LIST|
GET",scope=~"resource|",le="0.1"}[30d]))
                      or
                      vector(0)
                    )
                    +
                    sum(increase(apiserver_request_duration_seconds_bucket{verb=~"LIST|
GET",scope="namespace",le="0.5"}[30d]))
                    +
                    sum(increase(apiserver_request_duration_seconds_bucket{verb=~"LIST|
GET",scope="cluster",le="5"}[30d]))
                  )
                ) +
                # errors
                sum(code:apiserver_request_total:increase30d{code=~"5.."} or vector(0))
              )
              /
              sum(code:apiserver_request_total:increase30d)
            labels:
              verb: all
            record: apiserver_request:availability30d
          - expr: |2
              1 - (
                sum(increase(apiserver_request_duration_seconds_count{job="kubernetes-
apiservers",verb=~"LIST|GET"}[30d]))
                -
                (
                  # too slow
                  (
                    sum(increase(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope=~"resource|",le="0.1"}[30d]))
```

```
                    or
                    vector(0)
                  )
                  +
                  sum(increase(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="namespace",le="0.5"}[30d]))
                  +
                  sum(increase(apiserver_request_duration_seconds_bucket{job="kubernetes-
apiservers",verb=~"LIST|GET",scope="cluster",le="5"}[30d]))
                )
                +
                # errors
                sum(code:apiserver_request_total:increase30d{verb="read",code=~"5.."} or
vector(0))
              )
              /
              sum(code:apiserver_request_total:increase30d{verb="read"})
            labels:
              verb: read
            record: apiserver_request:availability30d
          - expr: |2
              1 - (
                (
                  # too slow
                  sum(increase(apiserver_request_duration_seconds_count{verb=~"POST|PUT|PATCH|
DELETE"}[30d]))
                  -
                  sum(increase(apiserver_request_duration_seconds_bucket{verb=~"POST|PUT|
PATCH|DELETE",le="1"}[30d]))
                )
                +
                # errors
                sum(code:apiserver_request_total:increase30d{verb="write",code=~"5.."} or
vector(0))
              )
              /
              sum(code:apiserver_request_total:increase30d{verb="write"})
            labels:
              verb: write
            record: apiserver_request:availability30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="LIST",code=~"2.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="GET",code=~"2.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="POST",code=~"2.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PUT",code=~"2.."}[30d]))
```

```
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PATCH",code=~"2.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="DELETE",code=~"2.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="LIST",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="GET",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="POST",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PUT",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PATCH",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="DELETE",code=~"3.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="LIST",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="GET",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="POST",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PUT",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PATCH",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
```

```
apiservers",verb="DELETE",code=~"4.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="LIST",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="GET",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="POST",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PUT",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="PATCH",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code, verb) (increase(apiserver_request_total{job="kubernetes-
apiservers",verb="DELETE",code=~"5.."}[30d]))
            record: code_verb:apiserver_request_total:increase30d
          - expr: |
              sum by (code) (code_verb:apiserver_request_total:increase30d{verb=~"LIST|GET"})
            labels:
              verb: read
            record: code:apiserver_request_total:increase30d
          - expr: |
              sum by (code) (code_verb:apiserver_request_total:increase30d{verb=~"POST|
PUT|PATCH|DELETE"})
            labels:
              verb: write
            record: code:apiserver_request_total:increase30d
    rules_yml: |
      {}
  deployment:
    configmapReload:
      containers:
        args:
          - --volume-dir=/etc/config
          - --webhook-url=http://127.0.0.1:9090/-/reload
    containers:
      args:
        - --storage.tsdb.retention.time=42d
        - --config.file=/etc/config/prometheus.yml
        - --storage.tsdb.path=/data
        - --web.console.libraries=/etc/prometheus/console_libraries
        - --web.console.templates=/etc/prometheus/consoles
        - --web.enable-lifecycle
    replicas: 1
    rollingUpdate:
```

```
      maxSurge: 25%
      maxUnavailable: 25%
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    storage: 20Gi
    storageClassName: default
  service:
    port: 80
    targetPort: 9090
    type: ClusterIP
pushgateway:
  deployment:
    replicas: 1
  service:
    port: 9091
    targetPort: 9091
    type: ClusterIP
```

## Configuración de Prometheus

La configuración de Prometheus se establece en el archivo `prometheus-data-values.yaml`. En la tabla se enumeran y se describen los parámetros disponibles.

### Tabla 11-5. Parámetros de configuración de Prometheus

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.namespace | Espacio de nombres en el que se implementará Prometheus | string | tanzu-system-monitoring |
| monitoring.create_namespace | La marca indica si se debe crear el espacio de nombres especificado por monitoring.namespace | booleano | false |
| monitoring.prometheus_server.config.prometheus_yaml | Detalles de configuración del clúster de Kubernetes que se pasarán a Prometheus | archivo yaml | prometheus.yaml |
| monitoring.prometheus_server.config.alerting_rules_yaml | Reglas de alerta detalladas definidas en Prometheus | archivo yaml | alerting_rules.yaml |
| monitoring.prometheus_server.config.recording_rules_yaml | Reglas de registro detalladas definidas en Prometheus | archivo yaml | recording_rules.yaml |
| monitoring.prometheus_server.service.type | Tipo de servicio para exponer Prometheus. Valores admitidos: ClusterIP | string | ClusterIP |
| monitoring.prometheus_server.enable_alerts.kubernetes_api | Habilitar alertas de SLO para la API de Kubernetes en Prometheus | booleano | true |

## Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.prometheus_server.sc.aws_type | Tipo de AWS definido para storageclass en AWS | string | gp2 |
| monitoring.prometheus_server.sc.aws_fsType | Tipo de sistema de archivos de AWS definido para storageclass en AWS | string | ext4 |
| monitoring.prometheus_server.sc.allowVolumeExpansion | Definir si se permite la expansión de volumen para storageclass en AWS | booleano | true |
| monitoring.prometheus_server.pvc.annotations | Anotaciones de clase de almacenamiento | mapa | {} |
| monitoring.prometheus_server.pvc.storage_class | Clase de almacenamiento que se utilizará para la notificación de volumen persistente. De forma predeterminada, es nulo y se utiliza el aprovisionador predeterminado. | string | nulo |
| monitoring.prometheus_server.pvc.accessMode | Defina el modo de acceso para la notificación de volumen persistente. Valores compatibles: ReadWriteOnce, ReadOnlyMany, ReadWriteMany | string | ReadWriteOnce |
| monitoring.prometheus_server.pvc.storage | Definir tamaño de almacenamiento para notificación de volumen persistente | string | 8Gi |
| monitoring.prometheus_server.deployment.replicas | Cantidad de réplicas de Prometheus | entero | 1 |
| monitoring.prometheus_server.image.repository | Ubicación del repositorio con la imagen de Prometheus. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.prometheus_server.image.name | Nombre de la imagen de Prometheus | string | prometheus |
| monitoring.prometheus_server.image.tag | Etiqueta de la imagen de Prometheus. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v2.17.1_vmware.1 |

## Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.prometheus_server.image.pullPolicy | Directiva de extracción de imágenes de Prometheus | string | IfNotPresent |
| monitoring.alertmanager.config.slack_demo | Configuración de notificaciones de Slack para Alertmanager | string | `slack_demo:`<br>`  name: slack_demo`<br>`  slack_configs:`<br>`  -`<br>`api_url: https://`<br>`hooks.slack.com`<br>`    channel:`<br>`'#alertmanager-`<br>`test'` |
| monitoring.alertmanager.config.email_receiver | Configuración de notificaciones de correo electrónico para Alertmanager | string | `email_receiver:`<br>`  name: email-`<br>`receiver`<br>`  email_configs:`<br>`  - to:`<br>`demo@tanzu.com`<br>`    send_resolved:`<br>`false`<br>`    from: from-`<br>`email@tanzu.com`<br>`    smarthost:`<br>`smtp.eample.com:25`<br>`    require_tls:`<br>`false` |
| monitoring.alertmanager.service.type | Tipo de servicio para exponer Alertmanager. Valores admitidos: ClusterIP | string | ClusterIP |
| monitoring.alertmanager.image.repository | Ubicación del repositorio con la imagen de Alertmanager. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.alertmanager.image.name | Nombre de la imagen de Alertmanager | string | alertmanager |
| monitoring.alertmanager.image.tag | Etiqueta de la imagen de Alertmanager. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v0.20.0_vmware.1 |
| monitoring.alertmanager.image.pullPolicy | Directiva de extracción de imágenes de Alertmanager | string | IfNotPresent |

## Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.alertmanager.pvc.annotations | Anotaciones de clase de almacenamiento | mapa | {} |
| monitoring.alertmanager.pvc.storage_class | Clase de almacenamiento que se utilizará para la notificación de volumen persistente. De forma predeterminada, es nulo y se utiliza el aprovisionador predeterminado. | string | nulo |
| monitoring.alertmanager.pvc.accessMode | Defina el modo de acceso para la notificación de volumen persistente. Valores compatibles: ReadWriteOnce, ReadOnlyMany, ReadWriteMany | string | ReadWriteOnce |
| monitoring.alertmanager.pvc.storage | Definir tamaño de almacenamiento para notificación de volumen persistente | string | 2Gi |
| monitoring.alertmanager.deployment.replicas | Cantidad de réplicas de Alertmanager | entero | 1 |
| monitoring.kube_state_metrics.image.repository | Repositorio que contiene la imagen de kube-state-metircs. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.kube_state_metrics.image.name | Nombre contiene la imagen kube-state-metircs | string | kube-state-metrics |
| monitoring.kube_state_metrics.image.tag | Etiqueta de la imagen de kube-state-metircs. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v1.9.5_vmware.1 |
| monitoring.kube_state_metrics.image.pullPolicy | directiva de extracción de imágenes de kube-state-metircs | string | IfNotPresent |
| monitoring.kube_state_metrics.deployment.replicas | Cantidad de réplicas de kube-state-metrics | entero | 1 |

## Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.node_exporter.image.repository | Repositorio que contiene la imagen de node-exporter. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.node_exporter.image.name | Nombre de la imagen de node-exporter | string | node-exporter |
| monitoring.node_exporter.image.tag | Etiqueta de la imagen de node-exporter. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v0.18.1_vmware.1 |
| monitoring.node_exporter.image.pullPolicy | directiva de extracción de imágenes de nodo-exporter | string | IfNotPresent |
| monitoring.node_exporter.hostNetwork | Si se establece en `hostNetwork: true`, el pod puede utilizar el espacio de nombres de red y los recursos de red del nodo. | booleano | false |
| monitoring.node_exporter.deployment.replicas | Cantidad de réplicas de node-exporter | entero | 1 |
| monitoring.pushgateway.image.repository | Repositorio que contiene la imagen de pushgateway. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.pushgateway.image.name | Nombre de la imagen de pushgateway | string | pushgateway |
| monitoring.pushgateway.image.tag | Etiqueta de la imagen de pushgateway. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v1.2.0_vmware.1 |
| monitoring.pushgateway.image.pullPolicy | Directiva de extracción de imágenes de pushgateway | string | IfNotPresent |
| monitoring.pushgateway.deployment.replicas | Cantidad de réplicas de pushgateway | entero | 1 |

## Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.cadvisor.image.repository | Repositorio que contiene la imagen de cadvisor. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/prometheus |
| monitoring.cadvisor.image.name | Nombre de la imagen de cadvisor | string | cadvisor |
| monitoring.cadvisor.image.tag | Etiqueta de la imagen de cadvisor. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v0.36.0_vmware.1 |
| monitoring.cadvisor.image.pullPolicy | Directiva de extracción de imágenes de cadvisor | string | IfNotPresent |
| monitoring.cadvisor.deployment.replicas | Cantidad de réplicas de cadvisor | entero | 1 |
| monitoring.ingress.enabled | Habilita/deshabilita la entrada para Prometheus y Alertmanager | booleano | false<br>Para utilizar la entrada, establezca este campo en `true` e implemente Contour. Para acceder a Prometheus, actualice sus `/etc/hosts` locales con una entrada que asigne `prometheus.system.tanzu` a una dirección IP del nodo de trabajo. |
| monitoring.ingress.virtual_host_fqdn | Nombre de host para acceder a Prometheus y Alertmanager | string | prometheus.system.tanzu |
| monitoring.ingress.prometheus_prefix | Prefijo de ruta de acceso para Prometheus | string | / |
| monitoring.ingress.alertmanager_prefix | Prefijo de ruta de acceso para Alertmanager | string | /alertmanager/ |

### Tabla 11-5. Parámetros de configuración de Prometheus (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.ingress.tlsCertificate.tls.crt | Certificado opcional para la entrada si desea utilizar su propio certificado TLS. De forma predeterminada, se genera un certificado autofirmado | string | Certificado generado |
| monitoring.ingress.tlsCertificate.tls.key | Clave privada de certificado opcional para la entrada si desea utilizar su propio certificado TLS. | string | Clave de certificado generada |

### Tabla 11-6. Campos configurables para Prometheus_Server Configmap

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| evaluation_interval | Frecuencia para evaluar reglas | duration | 1m |
| scrape_interval | Frecuencia de extracción de destinos | duration | 1m |
| scrape_timeout | Tiempo agotado para una solicitud de extracción | duration | 10s |
| rule_files | Los archivos de reglas especifican una lista de globs. Las reglas y las alertas se leen desde todos los archivos que coincidan | archivo yaml | |
| scrape_configs | Una lista de configuraciones de extracción. | lista | |
| job_name | El nombre del trabajo asignado a métricas recopiladas de forma predeterminada | string | |
| kubernetes_sd_configs | Lista de configuraciones de detección de servicios de Kubernetes. | lista | |
| relabel_configs | Lista de configuraciones de reetiqueta de destino. | lista | |
| action | Acción que se realizará en función de la coincidencia de expresión regular. | string | |
| regex | Expresión regular con la que coincide el valor extraído. | string | |

## Tabla 11-6. Campos configurables para Prometheus_Server Configmap (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| source_labels | Las etiquetas de origen seleccionan valores de las etiquetas existentes. | string | |
| scheme | Configura el esquema de protocolo utilizado para las solicitudes. | string | |
| tls_config | Configura las opciones de TLS de la solicitud de chat. | string | |
| ca_file | Certificado de CA con el que se validará el certificado del servidor de API. | filename | |
| insecure_skip_verify | Deshabilite la validación del certificado del servidor. | booleano | |
| bearer_token_file | Información opcional de autenticación del archivo de token de portador. | filename | |
| replacement | Valor de reemplazo contra el que se realiza un reemplazo de expresión regular si la expresión regular coincide. | string | |
| target_label | Etiqueta en la que el valor resultante se escribe en una acción de reemplazo. | string | |

## Tabla 11-7. Campos configurables para Alertmanager Configmap

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| resolve_timeout | ResolveTimeout es el valor predeterminado que utiliza Alertmanager si la alerta no incluye EndsAt | duration | 5m |
| smtp_smarthost | El host SMTP a través del cual se envían los correos electrónicos. | duration | 1m |
| slack_api_url | URL de webhook de Slack. | string | global.slack_api_url |
| pagerduty_url | La URL de pagerduty a la que se enviarán solicitudes de API. | string | global.pagerduty_url |
| plantillas | Archivos desde los que se leen las definiciones de plantillas de notificación personalizadas | ruta del archivo | |

Tabla 11-7. Campos configurables para Alertmanager Configmap (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| group_by | agrupar las alertas por etiqueta | string | |
| group_interval | Establezca el tiempo de espera antes de enviar una notificación sobre nuevas alertas que se agregan a un grupo | duration | 5m |
| group_wait | Tiempo que se debe esperar inicialmente para enviar una notificación para un grupo de alertas | duration | 30s |
| repeat_interval | Tiempo de espera antes de volver a enviar una notificación si ya se ha enviado correctamente para una alerta | duration | 4h |
| receivers | Una lista de receptores de notificaciones. | lista | |
| severity | Gravedad del incidente. | string | |
| channel | El canal o el usuario al que se enviarán notificaciones. | string | |
| html | El cuerpo HTML de la notificación por correo electrónico. | string | |
| text | El cuerpo de texto de la notificación por correo electrónico. | string | |
| send_resolved | Indica si se informa sobre las alertas resueltas. | filename | |
| email_configs | Configuraciones para la integración del correo electrónico | booleano | |

Las anotaciones en los pods permiten un control preciso del proceso de extracción. Estas anotaciones deben formar parte de los metadatos del pod. No tendrán efecto si se establecen en otros objetos, como Services o DaemonSets.

Tabla 11-8. Anotaciones del pod de Prometheus

| Anotación del pod | Descripción |
| --- | --- |
| prometheus.io/scrape | La configuración predeterminada extraerá en todos los pods y, si se establece en false, esta anotación excluirá el pod del proceso de extracción. |
| prometheus.io/path | Si la ruta de las métricas no es /metrics, defínala con esta anotación. |
| prometheus.io/port | Realice la extracción del pod en el puerto indicado en lugar de en los puertos declarados del pod (el valor predeterminado es un destino sin puertos si no se declara ninguno). |

El siguiente manifiesto de DaemonSet indicará a Prometheus que realiza la extracción de todos sus pods en el puerto 9102.

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: weave
  labels:
    app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '9102'
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: gcr.io/google-containers/fluentd-elasticsearch:1.20
```

# Referencia del paquete de Grafana

En este tema se proporciona información de referencia para el paquete Grafana.

## Acerca de Grafana

Grafana (https://grafana.com/) es un software de visualización y análisis de código abierto. Grafana permite consultar, visualizar, alertar y explorar métricas independientemente de dónde se almacenen. Permite crear gráficos y visualizaciones a partir de los datos de aplicaciones.

Para instalar el paquete de Grafana en un clúster de TKG, consulte los siguientes temas:

- TKr para vSphere 8.x: Instalar Grafana
- TKr para vSphere 7.x: #unique_177

## Componentes del paquete de Grafana

El paquete Grafana se instala en el clúster el contenedor enumerado en la tabla. El paquete de Grafana extrae el contenedor del registro público especificado en el repositorio de paquetes.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
| --- | --- | --- | --- |
| Grafana | Implementación | 2 | Visualización de datos |

## Valores de datos de Grafana

A continuación se muestra un archivo de ejemplo `grafana-data-values.yaml` con las siguientes personalizaciones:

- La entrada está habilitada (ingress: enabled: true).
- La entrada está configurada para las URL que terminan en / (prefix:).
- El FQDN de Grafana es grafana.system.tanzu (virtual_host_fqdn:)
- La PVC para Grafana es de 2 GB y se creará bajo la storageClass de vSphere predeterminada
- La contraseña de administrador (codificada en base64) para la interfaz de usuario de Grafana (grafana: secret: admin_password:).

```
namespace: grafana-dashboard
grafana:
  deployment:
    replicas: 1
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    storage: 2Gi
    storageClassName: default
  secret:
    admin_password: admin
    admin_user: YWRtaW4=
    type: Opaque
  service:
    port: 80
    targetPort: 3000
    type: LoadBalancer
ingress:
  enabled: true
  prefix: /
  servicePort: 80
  virtual_host_fqdn: grafana.system.tanzu
```

## Configuración de Grafana

La configuración de Grafana se establece en `grafana-data-values.yaml`. En la tabla se enumeran y se describen los parámetros disponibles.

### Tabla 11-9. Parámetros de configuración de Grafana

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.namespace | Espacio de nombres en el que se implementará Prometheus | string | tanzu-system-monitoring |
| monitoring.create_namespace | La marca indica si se debe crear el espacio de nombres especificado por monitoring.namespace | booleano | false |
| monitoring.grafana.cluster_role.apiGroups | grupo de API definido para grafana clusterrole | lista | [""] |
| monitoring.grafana.cluster_role.resources | recursos definidos para grafana clusterrole | lista | ["configmaps", "secrets"] |
| monitoring.grafana.cluster_role.verbs | permiso de acceso definido para clusterrole | lista | ["get", "watch", "list"] |
| monitoring.grafana.config.grafana_ini | Detalles del archivo de configuración de Grafana | archivo de configuración | grafana.ini<br>En este archivo, la URL `grafana_net` se utiliza para integrar con Grafana. Por ejemplo, para importar el panel de control directamente desde Grafana.com. |
| monitoring.grafana.config.datasource.type | Tipo de origen de datos de Grafana | string | prometheus |
| monitoring.grafana.config.datasource.access | modo de acceso. proxy o directo (servidor o navegador en la interfaz de usuario) | string | proxy |
| monitoring.grafana.config.datasource.isDefault | marcar como origen de datos predeterminado de Grafana | booleano | true |
| monitoring.grafana.config.provider_yaml | Archivo de configuración para definir el proveedor del panel de control de grafana | archivo yaml | provider.yaml |
| monitoring.grafana.service.type | Tipo de servicio para exponer Grafana. Valores admitidos: ClusterIP, NodePort y LoadBalancer | string | vSphere: NodePort, aws/azure: LoadBalancer |

## Tabla 11-9. Parámetros de configuración de Grafana (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.grafana.pvc.storage_class | Defina el modo de acceso para la notificación de volumen persistente. Valores compatibles: ReadWriteOnce, ReadOnlyMany, ReadWriteMany | string | ReadWriteOnce |
| monitoring.grafana.pvc.storage | Definir tamaño de almacenamiento para notificación de volumen persistente | string | 2Gi |
| monitoring.grafana.deployment.replicas | Cantidad de réplicas de Grafana | entero | 1 |
| monitoring.grafana.image.repository | Ubicación del repositorio con la imagen de Grafana. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/grafana |
| monitoring.grafana.image.name | Nombre de la imagen de Grafana | string | grafana |
| monitoring.grafana.image.tag | Etiqueta de la imagen de Grafana. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | v7.3.5_vmware.1 |
| monitoring.grafana.image.pullPolicy | Directiva de extracción de imágenes de Grafana | string | IfNotPresent |
| monitoring.grafana.secret.type | Tipo de secreto definido para el panel de control de Grafana | string | Opaco |
| monitoring.grafana.secret.admin_user | nombre de usuario para acceder al panel de control de Grafana | string | YWRtaW4=<br>El valor tiene codificación base64; para descodificar: `echo "xxxxxx" \| base64 --decode` |
| monitoring.grafana.secret.admin_password | contraseña para acceder al panel de control de Grafana | string | nulo |
| monitoring.grafana.secret.ldap_toml | Si utiliza la autenticación LDAP, la ruta del archivo de configuración LDAP | string | "" |

### Tabla 11-9. Parámetros de configuración de Grafana (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
| --- | --- | --- | --- |
| monitoring.grafana_init_container.image.repository | Repositorio que contiene una imagen de contenedor de init de Grafana. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/grafana |
| monitoring.grafana_init_container.image.name | Nombre de la imagen de contenedor de init de Grafana | string | k8s-sidecar |
| monitoring.grafana_init_container.image.tag | Etiqueta de la imagen de contenedor de init de Grafana. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | 0.1.99 |
| monitoring.grafana_init_container.image.pullPolicy | directiva de extracción de imágenes de contenedor de init de Grafana | string | IfNotPresent |
| monitoring.grafana_sc_dashboard.image.repository | Repositorio que contiene la imagen del panel de control de Grafana. El valor predeterminado es el registro de VMware público. Cambie este valor si utiliza un repositorio privado (p. ej., un entorno aislado). | string | projects.registry.vmware.com/tkg/grafana |
| monitoring.grafana_sc_dashboard.image.name | Nombre de la imagen del panel de control de Grafana | string | k8s-sidecar |
| monitoring.grafana_sc_dashboard.image.tag | Etiqueta de la imagen del panel de control de Grafana. Es posible que este valor tenga que actualizarse si va a actualizar la versión. | string | 0.1.99 |
| monitoring.grafana_sc_dashboard.image.pullPolicy | directiva de extracción de imagen del panel de control de Grafana | string | IfNotPresent |
| monitoring.grafana.ingress.enabled | Habilita/inhabilita la entrada para Grafana | booleano | true |
| monitoring.grafana.ingress.virtual_host_fqdn | Nombre de host para acceder a Grafana | string | grafana.system.tanzu |

Tabla 11-9. Parámetros de configuración de Grafana (continuación)

| Parámetro | Descripción | Tipo | Predeterminado |
|---|---|---|---|
| monitoring.grafana.ingress.prefix | Prefijo de la ruta de acceso para Grafana | string | / |
| monitoring.grafana.ingress.tlsCertificate.tls.crt | Certificado opcional para la entrada si desea utilizar su propio certificado TLS. De forma predeterminada, se genera un certificado autofirmado | string | Certificado generado |
| monitoring.grafana.ingress.tlsCertificate.tls.key | Clave privada de certificado opcional para la entrada si desea utilizar su propio certificado TLS. | string | Clave de certificado generada |

# Referencia del paquete de Harbor

En este tema se proporciona información de referencia del paquete de Registro de Harbor.

## Acerca de Registro de Harbor

Harbor (https://goharbor.io/) es un sistema de registro de contenedor de código abierto que proporciona un repositorio de imágenes, análisis de vulnerabilidad de imágenes y administración de proyectos.

Para instalar el paquete de Harbor en un clúster de TKG en Supervisor, consulte los siguientes temas:

- TKr para vSphere 8.x: Instalar Registro de Harbor
- TKr para vSphere 7.x: #unique_178

## Componentes de Harbor

El paquete Harbor se instala en el clúster los contenedores enumerados en la tabla. El paquete extrae los contenedores del registro público especificado en el repositorio de paquetes.

| Contenedor | Tipo de recurso | Réplicas | Descripción |
|---|---|---|---|
| `harbor-core` | Implementación | 1 | Servidor de administración y configuración para Envoy |
| `harbor-database` | Pod | 1 | Base de datos de Postgres |
| `harbor-jobservice` | Implementación | 1 | Servicio de trabajo de Harbor |
| `harbor-notary-server` | Implementación | 1 | Servicio notarial de Harbor |
| `harbor-notary-signer` | Implementación | 1 | Notary de Harbor |
| `harbor-portal` | Implementación | 1 | Interfaz web de Harbor |
| `harbor-redis` | Pod | 1 | Instancia de Redis de Harbor |

| Contenedor | Tipo de recurso | Réplicas | Descripción |
| --- | --- | --- | --- |
| `harbor-registry` | Implementación | 2 | Instancia de registro de contenedor de Harbor |
| `harbor-trivy` | Pod | 1 | Escáner de vulnerabilidad de imagen de Harbor |

## Valores de datos de Harbor

A continuación se muestra un ejemplo de `harbor-data-values` para instalar Harbor.

| Valor de datos | Descripción |
| --- | --- |
| `hostname: myharbordomain.com` | El FQDN para acceder a la UI administrativa de Harbor y al servicio de registro. |
| `harborAdminPassword: change-it` | La contraseña inicial de la cuenta de administrador de Harbor. Esto solo se aplica durante la instalación. Puede actualizarla mediante la API o la interfaz de usuario de Harbor después de la instalación. |
| `secretKey: 0123456789ABCDEF` | La clave secreta utilizada para el cifrado. Debe ser una cadena de 16 caracteres. |
| `database.password: change-it` | La contraseña inicial de la base de datos de postgres. |
| `core.secret: change-it` | El secreto se utiliza cuando el servidor principal se comunica con otros componentes. |
| `xsrfKey: 0123456789ABCDEF0123456789ABCDEF` | La clave XSRF. Debe ser una cadena de 32 caracteres. |
| `jobservice.secret: change-it` | El secreto se utiliza cuando el servicio de trabajo se comunica con otros componentes. |
| `registry.secret: change-it` | El secreto se utiliza para proteger el estado de carga del back-end de almacenamiento del cliente y del registro. |
| `persistence.persistentVolumeClaim.registry.storageClass: mystorageclass` | Especifique la directiva de almacenamiento de vSphere utilizada para aprovisionar el volumen. |
| `persistence.persistentVolumeClaim.jobservice.storageClass: mystorageclass` | Especifique la directiva de almacenamiento de vSphere utilizada para aprovisionar el volumen. |
| `persistence.persistentVolumeClaim.database.storageClass: mystorageclass` | Especifique la directiva de almacenamiento de vSphere utilizada para aprovisionar el volumen. |
| `persistence.persistentVolumeClaim.redis.storageClass: mystorageclass` | Especifique la directiva de almacenamiento de vSphere utilizada para aprovisionar el volumen. |
| `persistence.persistentVolumeClaim.trivy.storageClass: mystorageclass` | Especifique la directiva de almacenamiento de vSphere utilizada para aprovisionar el volumen. |

## Configuración de Harbor

La configuración de Harbor se establece en el archivo `harbor-data-values.yaml`. En la tabla, se enumeran y describen los campos mínimos que se requieren para la implementación.

| Propiedad | Valor | Descripción |
| --- | --- | --- |
| hostname | FQDN | El FQDN que ha designado para acceder a la interfaz de usuario de Harbor y para hacer referencia al registro en las aplicaciones cliente. El dominio debe configurarse en un servidor DNS externo para que se resuelva en la IP del servicio Envoy creada por Contour. |
| tlsCertificate.tlsSecretLabels | {"managed-by": "vmware-vRegistry"} | El certificado que Tanzu Kubernetes Grid utiliza para instalar el certificado de CA de Harbor como raíz de confianza en clústeres de Tanzu Kubernetes Grid. |
| persistence.persistentVolumeClaim.registry.storageClass | Un nombre de directiva de almacenamiento. | Una clase de almacenamiento que se utiliza para las PVC del registro de Harbor. |
| persistence.persistentVolumeClaim.jobservice.storageClass | Un nombre de directiva de almacenamiento. | Una clase de almacenamiento que se utiliza para las PVC del servicio de trabajo de Harbor. |
| persistence.persistentVolumeClaim.database.storageClass | Un nombre de directiva de almacenamiento. | Una clase de almacenamiento que se utiliza para las PVC de la base de datos de Harbor. |
| persistence.persistentVolumeClaim.redis.storageClass | Un nombre de directiva de almacenamiento. | Una clase de almacenamiento que se utiliza para las PVC de Redis de Harbor. |
| persistence.persistentVolumeClaim.trivy.storageClass | Un nombre de directiva de almacenamiento. | Una clase de almacenamiento que se utiliza para las PVC de Trivy de Harbor. |

# Instalar paquetes estándar en un clúster de TKG mediante TKr para vSphere 7.x

Consulte esta sección para instalar los paquetes estándar admitidos en clústeres de Servicio TKG aprovisionados compatibles con TKr para vSphere 7.x.

## Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x

En esta sección se proporcionan instrucciones para instalar paquetes estándar en clústeres de TKG aprovisionados con TKr para vSphere 7.x.

### Requisitos

Estas instrucciones se validan con TKr versión 1.27.10 para vSphere 7.0.3.6 y TKr versión 1.27.10 para vSphere 8.0.1.1. En el momento de la publicación, este era el TKr más reciente disponible para vSphere 7.x. Los TKr para vSphere 7.x se pueden ejecutar en vSphere 8.x con el fin de actualizar de vSphere 7.x a vSphere 8.x.

Cumpla los siguientes requisitos previos:

- Administración de cargas de trabajo habilitada
- Supervisor implementado
- espacio de nombres de vSphere creado

  Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.
- Cliente de Linux con Herramientas de la CLI de Kubernetes para vSphere instalado

  Consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

**Nota** Si utiliza un clúster de TKG aprovisionado con una TKr para vSphere 8.x, consulte la siguiente documentación para obtener instrucciones de instalación de paquetes estándar: Instalar paquetes estándar en un clúster de TKG mediante TKr para vSphere 8.x. Para obtener más información sobre las versiones de TKr, consulte las notas de la versión.

## Crear un clúster de TKG

Cree un clúster de TKG para alojar paquetes estándar.

1 Cree un clúster de TKG.

  Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

  Ejemplo de especificación de clúster para la edición Photon de TKr v1.27.10.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-photon
  namespace: tkgs-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vsan-esa-default-policy-raid5
      tkr:
        reference:
          name: v1.27.10---vmware.1-fips.1-tkg.1 #TKR for v7
    nodePools:
    - name: worker
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vsan-esa-default-policy-raid5
  settings:
    storage:
      defaultClass: vsan-esa-default-policy-raid5
```

Ejemplo de especificación de clúster para la edición de Ubuntu de TKr v1.27.10.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-ubuntu
  namespace: tkgs-ns
  annotations:
    run.tanzu.vmware.com/resolve-os-image: os-name=ubuntu
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vsan-esa-default-policy-raid5
      tkr:
        reference:
          name: v1.27.10---vmware.1-fips.1-tkg.1.ubuntu #TKR for v7
    nodePools:
    - name: worker
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vsan-esa-default-policy-raid5
  settings:
    storage:
      defaultClass: vsan-esa-default-policy-raid5
```

## Instalar el administrador de paquete de Carvel

Instale el administrador de paquete de Carvel.

1 Inicie sesión en el clúster de TKG.

```
kubectl vsphere login --server=IP-or-FQDN --vsphere-username USER@vsphere.local --tanzu-kubernetes-cluster-name tkgs-cluster-photon --tanzu-kubernetes-cluster-namespace tkgs-ns
```

2 Instale el administrador de paquete de Carvel.

```
wget -O- https://carvel.dev/install.sh > install.sh
```

```
sed -i 's/wget -nv -O-/wget --no-check-certificate -nv -O-/' install.sh
```

```
sudo bash install.sh
```

3 Compruebe la instalación.

```
imgpkg version
```

## Instalar controladora Kapp

Consulte Instalar la controladora Kapp en TKr para vSphere 7.x.

## Agregar un repositorio de paquetes

Agregue la versión del repositorio de paquetes deseada.

1 Enumere la etiqueta de repositorio más reciente.

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/repo
```

2 Cree `packagerepo.yaml`.

Actualice la versión del repositorio para que coincida con la versión de destino.

```
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageRepository
metadata:
  name: tanzu-standard
  namespace: tkg-system
spec:
  fetch:
    imgpkgBundle:
      image: projects.registry.vmware.com/tkg/packages/standard/repo:v2024.2.1
```

3 Instale el repositorio de paquetes.

```
kubectl apply -f packagerepo.yaml
```

Resultado esperado:

```
packagerepository.packaging.carvel.dev/tanzu-standard created
```

4 Compruebe el repositorio de paquetes.

```
kubectl get packagerepositories -A
```

Resultado esperado:

```
NAMESPACE    NAME              AGE    DESCRIPTION
tkg-system   tanzu-standard    3m9s   Reconcile succeeded
```

## Instalar administrador de certificados

Consulte Instalar el administrador de certificados en TKr para vSphere 7.x.

## Instalar Contour con Envoy

Consulte Instalar Contour en TKr para vSphere 7.x.

## Instalar ExternalDNS

Consulte Instalar ExternalDNS en TKr para vSphere 7.x.

## Instalar Fluent Bit para el reenvío de registros

Consulte Instalar Fluent Bit en TKr para vSphere 7.x.

### Instalar Prometheus

Consulte Instalar Prometheus en TKr para vSphere 7.x.

### Instalar Grafana

Consulte Instalar Grafana en TKr para vSphere 7.x.

### Instalar Harbor

Consulte Instalar Harbor en TKr para vSphere 7.x.

## Instalar la controladora Kapp en TKr para vSphere 7.x

Consulte estas instrucciones para instalar la controladora Kapp en un clúster de TKG aprovisionado con TKr para vSphere 7.x.

### Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

### Instalar controladora Kapp

**Importante** Estas instrucciones son específicas para TKr para vSphere 7.x. Los TKr para vSphere 8.x ya incluye el paquete de la controladora Kapp. No instale manualmente la controladora Kapp en TKr para vSphere 8.x.

Instale la controladora Kapp.

1 Cree un enlace para ejecutar el pod de la controladora Kapp.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=cluster-admin --group=system:authenticated
```

Resultado esperado:

```
clusterrolebinding.rbac.authorization.k8s.io/default-tkg-admin-privileged-binding created
```

2 Prepare `kapp-controller.yaml`.

Consulte

3 Instale la controladora Kapp.

```
kubectl apply -f kapp-controller.yaml
```

4 Compruebe la instalación de la controladora Kapp.

```
kubectl get all -n tkg-system
```

Resultado de ejemplo:

```
NAME                                 READY   STATUS    RESTARTS   AGE
pod/kapp-controller-b7576ddd-p8s87   2/2     Running   0          5m33s
NAME                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/packaging-api   ClusterIP   198.201.96.77   <none>        443/TCP   5m34s
NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kapp-controller   1/1     1            1           5m33s
```

5 Compruebe el recurso personalizado de Carvel.

```
kubectl get crd | grep carvel
```

Resultado de ejemplo:

```
internalpackagemetadatas.internal.packaging.carvel.dev   2024-03-12T08:27:21Z
internalpackages.internal.packaging.carvel.dev           2024-03-12T08:27:21Z
packageinstalls.packaging.carvel.dev                     2024-03-12T08:27:21Z
packagerepositories.packaging.carvel.dev                 2024-03-12T08:27:22Z
```

## kapp-controller.yaml

El siguiente archivo `kapp-controller.yaml` incluye la configuración de `securityContext` requerida.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: tkg-system
---
apiVersion: v1
kind: Namespace
metadata:
  name: kapp-controller-packaging-global
---
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  name: v1alpha1.data.packaging.carvel.dev
spec:
  group: data.packaging.carvel.dev
  groupPriorityMinimum: 100
  service:
    name: packaging-api
    namespace: tkg-system
  version: v1alpha1
  versionPriority: 100
---
apiVersion: v1
kind: Service
metadata:
  name: packaging-api
  namespace: tkg-system
```

```
spec:
  ports:
  - port: 443
    protocol: TCP
    targetPort: api
  selector:
    app: kapp-controller
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: internalpackagemetadatas.internal.packaging.carvel.dev
spec:
  group: internal.packaging.carvel.dev
  names:
    kind: InternalPackageMetadata
    listKind: InternalPackageMetadataList
    plural: internalpackagemetadatas
    singular: internalpackagemetadata
  scope: Namespaced
  versions:
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
              submits requests to. Cannot be updated. In CamelCase. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            properties:
              categories:
                description: Classifiers of the package (optional; Array of strings)
                items:
                  type: string
                type: array
              displayName:
                description: Human friendly name of the package (optional; string)
                type: string
              iconSVGBase64:
                description: Base64 encoded icon (optional; string)
                type: string
              longDescription:
                description: Long description of the package (optional; string)
                type: string
```

```
              maintainers:
                description: List of maintainer info for the package. Currently only
                  supports the name key. (optional; array of maintner info)
                items:
                  properties:
                    name:
                      type: string
                  type: object
                type: array
              providerName:
                description: Name of the entity distributing the package (optional;
                  string)
                type: string
              shortDescription:
                description: Short desription of the package (optional; string)
                type: string
              supportDescription:
                description: Description of the support available for the package
                  (optional; string)
                type: string
            type: object
        required:
        - spec
        type: object
    served: true
    storage: true
    subresources:
      status: {}
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: internalpackages.internal.packaging.carvel.dev
spec:
  group: internal.packaging.carvel.dev
  names:
    kind: InternalPackage
    listKind: InternalPackageList
    plural: internalpackages
    singular: internalpackage
  scope: Namespaced
  versions:
  - name: v1alpha1
    schema:
      openAPIV3Schema:
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
```

```
              submits requests to. Cannot be updated. In CamelCase. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            properties:
              capacityRequirementsDescription:
                description: 'System requirements needed to install the package. Note:
                  these requirements will not be verified by kapp-controller on installation.
                  (optional; string)'
                type: string
              includedSoftware:
                description: IncludedSoftware can be used to show the software contents
                  of a Package. This is especially useful if the underlying versions
                  do not match the Package version
                items:
                  description: IncludedSoftware contains the underlying Software Contents
                    of a Package
                  properties:
                    description:
                      type: string
                    displayName:
                      type: string
                    version:
                      type: string
                  type: object
                type: array
              kappControllerVersionSelection:
                description: KappControllerVersionSelection specifies the versions
                  of kapp-controller which can install this package
                properties:
                  constraints:
                    type: string
                type: object
              kubernetesVersionSelection:
                description: KubernetesVersionSelection specifies the versions of
                  k8s which this package can be installed on
                properties:
                  constraints:
                    type: string
                type: object
              licenses:
                description: Description of the licenses that apply to the package
                  software (optional; Array of strings)
                items:
                  type: string
                type: array
              refName:
                description: The name of the PackageMetadata associated with this
                  version Must be a valid PackageMetadata name (see PackageMetadata
                  CR for details) Cannot be empty
                type: string
              releaseNotes:
                description: Version release notes (optional; string)
```

```
        type: string
      releasedAt:
        description: Timestamp of release (iso8601 formatted string; optional)
        format: date-time
        nullable: true
        type: string
      template:
        properties:
          spec:
            properties:
              canceled:
                description: Cancels current and future reconciliations (optional;
                  default=false)
                type: boolean
              cluster:
                description: Specifies that app should be deployed to destination
                  cluster; by default, cluster is same as where this resource
                  resides (optional; v0.5.0+)
                properties:
                  kubeconfigSecretRef:
                    description: Specifies secret containing kubeconfig (required)
                    properties:
                      key:
                        description: Specifies key that contains kubeconfig
                          (optional)
                        type: string
                      name:
                        description: Specifies secret name within app's namespace
                          (required)
                        type: string
                    type: object
                  namespace:
                    description: Specifies namespace in destination cluster
                      (optional)
                    type: string
                type: object
              deploy:
                items:
                  properties:
                    kapp:
                      description: Use kapp to deploy resources
                      properties:
                        delete:
                          description: Configuration for delete command (optional)
                          properties:
                            rawOptions:
                              description: Pass through options to kapp delete
                                (optional)
                              items:
                                type: string
                              type: array
                          type: object
                        inspect:
                          description: 'Configuration for inspect command
                            (optional) as of kapp-controller v0.31.0, inspect
```

```
                    is disabled by default add rawOptions or use an
                    empty inspect config like `inspect: {}` to enable'
                  properties:
                    rawOptions:
                      description: Pass through options to kapp inspect
                        (optional)
                      items:
                        type: string
                      type: array
                  type: object
                intoNs:
                  description: Override namespace for all resources
                    (optional)
                  type: string
                mapNs:
                  description: Provide custom namespace override mapping
                    (optional)
                  items:
                    type: string
                  type: array
                rawOptions:
                  description: Pass through options to kapp deploy
                    (optional)
                  items:
                    type: string
                  type: array
              type: object
          type: object
        type: array
      fetch:
        items:
          properties:
            git:
              description: Uses git to clone repository
              properties:
                lfsSkipSmudge:
                  description: Skip lfs download (optional)
                  type: boolean
                ref:
                  description: Branch, tag, commit; origin is the
                    name of the remote (optional)
                  type: string
                refSelection:
                  description: Specifies a strategy to resolve to
                    an explicit ref (optional; v0.24.0+)
                  properties:
                    semver:
                      properties:
                        constraints:
                          type: string
                        prereleases:
                          properties:
                            identifiers:
                              items:
                                type: string
```

```
         type: array
       type: object
     type: object
   type: object
  secretRef:
   description: 'Secret with auth details. allowed
    keys: ssh-privatekey, ssh-knownhosts, username,
    password (optional) (if ssh-knownhosts is not
    specified, git will not perform strict host checking)'
   properties:
    name:
     description: Object is expected to be within
      same namespace
     type: string
   type: object
  subPath:
   description: Grab only portion of repository (optional)
   type: string
  url:
   description: http or ssh urls are supported (required)
   type: string
 type: object
helmChart:
 description: Uses helm fetch to fetch specified chart
 properties:
  name:
   description: 'Example: stable/redis'
   type: string
  repository:
   properties:
    secretRef:
     properties:
      name:
       description: Object is expected to be within
        same namespace
       type: string
     type: object
    url:
     description: Repository url; scheme of oci://
      will fetch experimental helm oci chart (v0.19.0+)
      (required)
     type: string
   type: object
  version:
   type: string
 type: object
http:
 description: Uses http library to fetch file
 properties:
  secretRef:
   description: 'Secret to provide auth details (optional)
    Secret may include one or more keys: username,
    password'
   properties:
    name:
```

```
          description: Object is expected to be within
            same namespace
          type: string
      type: object
    sha256:
      description: Checksum to verify after download (optional)
      type: string
    subPath:
      description: Grab only portion of download (optional)
      type: string
    url:
      description: 'URL can point to one of following
        formats: text, tgz, zip http and https url are
        supported; plain file, tgz and tar types are supported
        (required)'
      type: string
  type: object
image:
  description: Pulls content from Docker/OCI registry
  properties:
    secretRef:
      description: 'Secret may include one or more keys:
        username, password, token. By default anonymous
        access is used for authentication.'
      properties:
        name:
          description: Object is expected to be within
            same namespace
          type: string
      type: object
    subPath:
      description: Grab only portion of image (optional)
      type: string
    tagSelection:
      description: Specifies a strategy to choose a tag
        (optional; v0.24.0+) if specified, do not include
        a tag in url key
      properties:
        semver:
          properties:
            constraints:
              type: string
            prereleases:
              properties:
                identifiers:
                  items:
                    type: string
                  type: array
              type: object
          type: object
      type: object
    url:
      description: 'Docker image url; unqualified, tagged,
        or digest references supported (required) Example:
        username/app1-config:v0.1.0'
```

```
      type: string
  type: object
imgpkgBundle:
  description: Pulls imgpkg bundle from Docker/OCI registry
    (v0.17.0+)
  properties:
    image:
      description: Docker image url; unqualified, tagged,
        or digest references supported (required)
      type: string
    secretRef:
      description: 'Secret may include one or more keys:
        username, password, token. By default anonymous
        access is used for authentication.'
      properties:
        name:
          description: Object is expected to be within
            same namespace
          type: string
      type: object
    tagSelection:
      description: Specifies a strategy to choose a tag
        (optional; v0.24.0+) if specified, do not include
        a tag in url key
      properties:
        semver:
          properties:
            constraints:
              type: string
            prereleases:
              properties:
                identifiers:
                  items:
                    type: string
                  type: array
              type: object
          type: object
      type: object
  type: object
inline:
  description: Pulls content from within this resource;
    or other resources in the cluster
  properties:
    paths:
      additionalProperties:
        type: string
      description: Specifies mapping of paths to their
        content; not recommended for sensitive values
        as CR is not encrypted (optional)
      type: object
    pathsFrom:
      description: Specifies content via secrets and config
        maps; data values are recommended to be placed
        in secrets (optional)
      items:
```

```
                  properties:
                    configMapRef:
                      properties:
                        directoryPath:
                          description: Specifies where to place
                            files found in secret (optional)
                          type: string
                        name:
                          type: string
                      type: object
                    secretRef:
                      properties:
                        directoryPath:
                          description: Specifies where to place
                            files found in secret (optional)
                          type: string
                        name:
                          type: string
                      type: object
                  type: object
                type: array
            type: object
          path:
            description: Relative path to place the fetched artifacts
            type: string
        type: object
      type: array
    noopDelete:
      description: Deletion requests for the App will result in
        the App CR being deleted, but its associated resources will
        not be deleted (optional; default=false; v0.18.0+)
      type: boolean
    paused:
      description: Pauses _future_ reconciliation; does _not_ affect
        currently running reconciliation (optional; default=false)
      type: boolean
    serviceAccountName:
      description: Specifies that app should be deployed authenticated
        via given service account, found in this namespace (optional;
        v0.6.0+)
      type: string
    syncPeriod:
      description: Specifies the length of time to wait, in time
        + unit format, before reconciling. Always >= 30s. If value
        below 30s is specified, 30s will be used. (optional; v0.9.0+;
        default=30s)
      type: string
    template:
      items:
        properties:
          cue:
            properties:
              inputExpression:
                description: Cue expression for single path component,
                  can be used to unify ValuesFrom into a given field
```

```
    (optional)
  type: string
outputExpression:
  description: Cue expression to output, default will
    export all visible fields (optional)
  type: string
paths:
  description: Explicit list of files/directories
    (optional)
  items:
    type: string
  type: array
valuesFrom:
  description: Provide values (optional)
  items:
    properties:
      configMapRef:
        properties:
          name:
            type: string
        type: object
      downwardAPI:
        properties:
          items:
            items:
              properties:
                fieldPath:
                  description: 'Required: Selects
                    a field of the app: only annotations,
                    labels, uid, name and namespace
                    are supported.'
                  type: string
                kappControllerVersion:
                  description: 'Optional: Get running
                    KappController version, defaults
                    (empty) to retrieving the current
                    running version.. Can be manually
                    supplied instead.'
                  properties:
                    version:
                      type: string
                  type: object
                kubernetesAPIs:
                  description: 'Optional: Get running
                    KubernetesAPIs from cluster, defaults
                    (empty) to retrieving the APIs
                    from the cluster. Can be manually
                    supplied instead, e.g ["group/version",
                    "group2/version2"]'
                  properties:
                    groupVersions:
                      items:
                        type: string
                      type: array
                  type: object
```

```
                    kubernetesVersion:
                      description: 'Optional: Get running
                        Kubernetes version from cluster,
                        defaults (empty) to retrieving
                        the version from the cluster.
                        Can be manually supplied instead.'
                      properties:
                        version:
                          type: string
                      type: object
                    name:
                      type: string
                  type: object
                type: array
            type: object
          path:
            type: string
          secretRef:
            properties:
              name:
                type: string
            type: object
        type: object
      type: array
  type: object
helmTemplate:
  description: Use helm template command to render helm
    chart
  properties:
    kubernetesAPIs:
      description: 'Optional: Use kubernetes group/versions
        resources available in the live cluster'
      properties:
        groupVersions:
          items:
            type: string
          type: array
      type: object
    kubernetesVersion:
      description: 'Optional: Get Kubernetes version,
        defaults (empty) to retrieving the version from
        the cluster. Can be manually overridden to a value
        instead.'
      properties:
        version:
          type: string
      type: object
    name:
      description: Set name explicitly, default is App
        CR's name (optional; v0.13.0+)
      type: string
    namespace:
      description: Set namespace explicitly, default is
        App CR's namespace (optional; v0.13.0+)
      type: string
```

```
                                path:
                                  description: Path to chart (optional; v0.13.0+)
                                  type: string
                                valuesFrom:
                                  description: One or more secrets, config maps, paths
                                    that provide values (optional)
                                  items:
                                    properties:
                                      configMapRef:
                                        properties:
                                          name:
                                            type: string
                                        type: object
                                      downwardAPI:
                                        properties:
                                          items:
                                            items:
                                              properties:
                                                fieldPath:
                                                  description: 'Required: Selects
                                                    a field of the app: only annotations,
                                                    labels, uid, name and namespace
                                                    are supported.'
                                                  type: string
                                                kappControllerVersion:
                                                  description: 'Optional: Get running
                                                    KappController version, defaults
                                                    (empty) to retrieving the current
                                                    running version.. Can be manually
                                                    supplied instead.'
                                                  properties:
                                                    version:
                                                      type: string
                                                  type: object
                                                kubernetesAPIs:
                                                  description: 'Optional: Get running
                                                    KubernetesAPIs from cluster, defaults
                                                    (empty) to retrieving the APIs
                                                    from the cluster. Can be manually
                                                    supplied instead, e.g ["group/version",
                                                    "group2/version2"]'
                                                  properties:
                                                    groupVersions:
                                                      items:
                                                        type: string
                                                      type: array
                                                  type: object
                                                kubernetesVersion:
                                                  description: 'Optional: Get running
                                                    Kubernetes version from cluster,
                                                    defaults (empty) to retrieving
                                                    the version from the cluster.
                                                    Can be manually supplied instead.'
                                                  properties:
                                                    version:
```

```
                          type: string
                      type: object
                    name:
                      type: string
                  type: object
                type: array
            type: object
          path:
            type: string
          secretRef:
            properties:
              name:
                type: string
            type: object
        type: object
      type: array
  type: object
jsonnet:
  description: TODO implement jsonnet
  type: object
kbld:
  description: Use kbld to resolve image references to
    use digests
  properties:
    paths:
      items:
        type: string
      type: array
  type: object
kustomize:
  description: TODO implement kustomize
  type: object
sops:
  description: Use sops to decrypt *.sops.yml files (optional;
    v0.11.0+)
  properties:
    age:
      properties:
        privateKeysSecretRef:
          description: Secret with private armored PGP
            private keys (required)
          properties:
            name:
              type: string
          type: object
      type: object
    paths:
      description: Lists paths to decrypt explicitly (optional;
        v0.13.0+)
      items:
        type: string
      type: array
    pgp:
      description: Use PGP to decrypt files (required)
      properties:
```

```
                                     privateKeysSecretRef:
                                       description: Secret with private armored PGP
                                         private keys (required)
                                       properties:
                                         name:
                                           type: string
                                       type: object
                                   type: object
                               type: object
                             ytt:
                               description: Use ytt to template configuration
                               properties:
                                 fileMarks:
                                   description: Control metadata about input files
                                     passed to ytt (optional; v0.18.0+) see https://
carvel.dev/ytt/docs/latest/file-marks/
                                     for more details
                                   items:
                                     type: string
                                   type: array
                                 ignoreUnknownComments:
                                   description: Ignores comments that ytt doesn't recognize
                                     (optional; default=false)
                                   type: boolean
                                 inline:
                                   description: Specify additional files, including
                                     data values (optional)
                                   properties:
                                     paths:
                                       additionalProperties:
                                         type: string
                                       description: Specifies mapping of paths to their
                                         content; not recommended for sensitive values
                                         as CR is not encrypted (optional)
                                       type: object
                                     pathsFrom:
                                       description: Specifies content via secrets and
                                         config maps; data values are recommended to
                                         be placed in secrets (optional)
                                       items:
                                         properties:
                                           configMapRef:
                                             properties:
                                               directoryPath:
                                                 description: Specifies where to place
                                                   files found in secret (optional)
                                                 type: string
                                               name:
                                                 type: string
                                             type: object
                                           secretRef:
                                             properties:
                                               directoryPath:
                                                 description: Specifies where to place
                                                   files found in secret (optional)
```

```
                                                type: string
                                              name:
                                                type: string
                                            type: object
                                        type: object
                                      type: array
                                  type: object
                                paths:
                                  description: Lists paths to provide to ytt explicitly
                                    (optional)
                                  items:
                                    type: string
                                  type: array
                                strict:
                                  description: Forces strict mode https://github.com/k14s/ytt/
blob/develop/docs/strict.md
                                    (optional; default=false)
                                  type: boolean
                                valuesFrom:
                                  description: Provide values via ytt's --data-values-file
                                    (optional; v0.19.0-alpha.9)
                                  items:
                                    properties:
                                      configMapRef:
                                        properties:
                                          name:
                                            type: string
                                        type: object
                                      downwardAPI:
                                        properties:
                                          items:
                                            items:
                                              properties:
                                                fieldPath:
                                                  description: 'Required: Selects
                                                    a field of the app: only annotations,
                                                    labels, uid, name and namespace
                                                    are supported.'
                                                  type: string
                                                kappControllerVersion:
                                                  description: 'Optional: Get running
                                                    KappController version, defaults
                                                    (empty) to retrieving the current
                                                    running version.. Can be manually
                                                    supplied instead.'
                                                  properties:
                                                    version:
                                                      type: string
                                                  type: object
                                                kubernetesAPIs:
                                                  description: 'Optional: Get running
                                                    KubernetesAPIs from cluster, defaults
                                                    (empty) to retrieving the APIs
                                                    from the cluster. Can be manually
                                                    supplied instead, e.g ["group/version",
```

```
                                            "group2/version2"]'
                                          properties:
                                            groupVersions:
                                              items:
                                                type: string
                                              type: array
                                          type: object
                                        kubernetesVersion:
                                          description: 'Optional: Get running
                                            Kubernetes version from cluster,
                                            defaults (empty) to retrieving
                                            the version from the cluster.
                                            Can be manually supplied instead.'
                                          properties:
                                            version:
                                              type: string
                                          type: object
                                        name:
                                          type: string
                                      type: object
                                    type: array
                                type: object
                              path:
                                type: string
                              secretRef:
                                properties:
                                  name:
                                    type: string
                                type: object
                            type: object
                          type: array
                      type: object
                  type: object
                type: array
            type: object
        required:
        - spec
        type: object
      valuesSchema:
        description: valuesSchema can be used to show template values that
          can be configured by users when a Package is installed in an OpenAPI
          schema format.
        properties:
          openAPIv3:
            nullable: true
            type: object
            x-kubernetes-preserve-unknown-fields: true
        type: object
      version:
        description: Package version; Referenced by PackageInstall; Must be
          valid semver (required) Cannot be empty
        type: string
    type: object
required:
- spec
```

```
        type: object
    served: true
    storage: true
    subresources:
      status: {}
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: apps.kappctrl.k14s.io
spec:
  group: kappctrl.k14s.io
  names:
    categories:
    - carvel
    kind: App
    listKind: AppList
    plural: apps
    singular: app
  scope: Namespaced
  versions:
  - additionalPrinterColumns:
    - description: Friendly description
      jsonPath: .status.friendlyDescription
      name: Description
      type: string
    - description: Last time app started being deployed. Does not mean anything was
        changed.
      jsonPath: .status.deploy.startedAt
      name: Since-Deploy
      type: date
    - description: Time since creation
      jsonPath: .metadata.creationTimestamp
      name: Age
      type: date
    name: v1alpha1
    schema:
      openAPIV3Schema:
        description: 'An App is a set of Kubernetes resources. These resources could
          span any number of namespaces or could be cluster-wide (e.g. CRDs). An App
          is represented in kapp-controller using a App CR. The App CR comprises of
          three main sections: spec.fetch â€" declare source for fetching configuration
          and OCI images spec.template â€" declare templating tool and values spec.deploy
          â€" declare deployment tool and any deploy specific configuration'
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
              submits requests to. Cannot be updated. In CamelCase. More info: https://
```

```
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            properties:
              canceled:
                description: Cancels current and future reconciliations (optional;
                  default=false)
                type: boolean
              cluster:
                description: Specifies that app should be deployed to destination
                  cluster; by default, cluster is same as where this resource resides
                  (optional; v0.5.0+)
                properties:
                  kubeconfigSecretRef:
                    description: Specifies secret containing kubeconfig (required)
                    properties:
                      key:
                        description: Specifies key that contains kubeconfig (optional)
                        type: string
                      name:
                        description: Specifies secret name within app's namespace
                          (required)
                        type: string
                    type: object
                  namespace:
                    description: Specifies namespace in destination cluster (optional)
                    type: string
                type: object
              deploy:
                items:
                  properties:
                    kapp:
                      description: Use kapp to deploy resources
                      properties:
                        delete:
                          description: Configuration for delete command (optional)
                          properties:
                            rawOptions:
                              description: Pass through options to kapp delete (optional)
                              items:
                                type: string
                              type: array
                          type: object
                        inspect:
                          description: 'Configuration for inspect command (optional)
                            as of kapp-controller v0.31.0, inspect is disabled by
                            default add rawOptions or use an empty inspect config
                            like `inspect: {}` to enable'
                          properties:
                            rawOptions:
                              description: Pass through options to kapp inspect (optional)
                              items:
                                type: string
```

```
                              type: array
                            type: object
                          intoNs:
                            description: Override namespace for all resources (optional)
                            type: string
                          mapNs:
                            description: Provide custom namespace override mapping (optional)
                            items:
                              type: string
                            type: array
                          rawOptions:
                            description: Pass through options to kapp deploy (optional)
                            items:
                              type: string
                            type: array
                        type: object
                    type: object
                  type: array
                fetch:
                  items:
                    properties:
                      git:
                        description: Uses git to clone repository
                        properties:
                          lfsSkipSmudge:
                            description: Skip lfs download (optional)
                            type: boolean
                          ref:
                            description: Branch, tag, commit; origin is the name of
                              the remote (optional)
                            type: string
                          refSelection:
                            description: Specifies a strategy to resolve to an explicit
                              ref (optional; v0.24.0+)
                            properties:
                              semver:
                                properties:
                                  constraints:
                                    type: string
                                  prereleases:
                                    properties:
                                      identifiers:
                                        items:
                                          type: string
                                        type: array
                                    type: object
                                type: object
                            type: object
                          secretRef:
                            description: 'Secret with auth details. allowed keys: ssh-
privatekey,
                              ssh-knownhosts, username, password (optional) (if ssh-knownhosts
                              is not specified, git will not perform strict host checking)'
                            properties:
                              name:
```

```
                  description: Object is expected to be within same namespace
                  type: string
                type: object
              subPath:
                description: Grab only portion of repository (optional)
                type: string
              url:
                description: http or ssh urls are supported (required)
                type: string
            type: object
          helmChart:
            description: Uses helm fetch to fetch specified chart
            properties:
              name:
                description: 'Example: stable/redis'
                type: string
              repository:
                properties:
                  secretRef:
                    properties:
                      name:
                        description: Object is expected to be within same
                          namespace
                        type: string
                    type: object
                  url:
                    description: Repository url; scheme of oci:// will fetch
                      experimental helm oci chart (v0.19.0+) (required)
                    type: string
                type: object
              version:
                type: string
            type: object
          http:
            description: Uses http library to fetch file
            properties:
              secretRef:
                description: 'Secret to provide auth details (optional)
                  Secret may include one or more keys: username, password'
                properties:
                  name:
                    description: Object is expected to be within same namespace
                    type: string
                type: object
              sha256:
                description: Checksum to verify after download (optional)
                type: string
              subPath:
                description: Grab only portion of download (optional)
                type: string
              url:
                description: 'URL can point to one of following formats:
                  text, tgz, zip http and https url are supported; plain
                  file, tgz and tar types are supported (required)'
                type: string
```

```
                    type: object
                  image:
                    description: Pulls content from Docker/OCI registry
                    properties:
                      secretRef:
                        description: 'Secret may include one or more keys: username,
                          password, token. By default anonymous access is used for
                          authentication.'
                        properties:
                          name:
                            description: Object is expected to be within same namespace
                            type: string
                        type: object
                      subPath:
                        description: Grab only portion of image (optional)
                        type: string
                      tagSelection:
                        description: Specifies a strategy to choose a tag (optional;
                          v0.24.0+) if specified, do not include a tag in url key
                        properties:
                          semver:
                            properties:
                              constraints:
                                type: string
                              prereleases:
                                properties:
                                  identifiers:
                                    items:
                                      type: string
                                    type: array
                                type: object
                            type: object
                        type: object
                      url:
                        description: 'Docker image url; unqualified, tagged, or
                          digest references supported (required) Example: username/app1-
config:v0.1.0'
                        type: string
                    type: object
                  imgpkgBundle:
                    description: Pulls imgpkg bundle from Docker/OCI registry (v0.17.0+)
                    properties:
                      image:
                        description: Docker image url; unqualified, tagged, or digest
                          references supported (required)
                        type: string
                      secretRef:
                        description: 'Secret may include one or more keys: username,
                          password, token. By default anonymous access is used for
                          authentication.'
                        properties:
                          name:
                            description: Object is expected to be within same namespace
                            type: string
                        type: object
```

```
            tagSelection:
              description: Specifies a strategy to choose a tag (optional;
                v0.24.0+) if specified, do not include a tag in url key
              properties:
                semver:
                  properties:
                    constraints:
                      type: string
                    prereleases:
                      properties:
                        identifiers:
                          items:
                            type: string
                          type: array
                      type: object
                  type: object
              type: object
          type: object
        inline:
          description: Pulls content from within this resource; or other
            resources in the cluster
          properties:
            paths:
              additionalProperties:
                type: string
              description: Specifies mapping of paths to their content;
                not recommended for sensitive values as CR is not encrypted
                (optional)
              type: object
            pathsFrom:
              description: Specifies content via secrets and config maps;
                data values are recommended to be placed in secrets (optional)
              items:
                properties:
                  configMapRef:
                    properties:
                      directoryPath:
                        description: Specifies where to place files found
                          in secret (optional)
                        type: string
                      name:
                        type: string
                    type: object
                  secretRef:
                    properties:
                      directoryPath:
                        description: Specifies where to place files found
                          in secret (optional)
                        type: string
                      name:
                        type: string
                    type: object
                type: object
              type: array
          type: object
```

```
                    path:
                      description: Relative path to place the fetched artifacts
                      type: string
                  type: object
                type: array
              noopDelete:
                description: Deletion requests for the App will result in the App
                  CR being deleted, but its associated resources will not be deleted
                  (optional; default=false; v0.18.0+)
                type: boolean
              paused:
                description: Pauses _future_ reconciliation; does _not_ affect currently
                  running reconciliation (optional; default=false)
                type: boolean
              serviceAccountName:
                description: Specifies that app should be deployed authenticated via
                  given service account, found in this namespace (optional; v0.6.0+)
                type: string
              syncPeriod:
                description: Specifies the length of time to wait, in time + unit
                  format, before reconciling. Always >= 30s. If value below 30s is
                  specified, 30s will be used. (optional; v0.9.0+; default=30s)
                type: string
              template:
                items:
                  properties:
                    cue:
                      properties:
                        inputExpression:
                          description: Cue expression for single path component, can
                            be used to unify ValuesFrom into a given field (optional)
                          type: string
                        outputExpression:
                          description: Cue expression to output, default will export
                            all visible fields (optional)
                          type: string
                        paths:
                          description: Explicit list of files/directories (optional)
                          items:
                            type: string
                          type: array
                        valuesFrom:
                          description: Provide values (optional)
                          items:
                            properties:
                              configMapRef:
                                properties:
                                  name:
                                    type: string
                                type: object
                              downwardAPI:
                                properties:
                                  items:
                                    items:
                                      properties:
```

```
                    fieldPath:
                      description: 'Required: Selects a field
                        of the app: only annotations, labels,
                        uid, name and namespace are supported.'
                      type: string
                    kappControllerVersion:
                      description: 'Optional: Get running KappController
                        version, defaults (empty) to retrieving
                        the current running version.. Can be manually
                        supplied instead.'
                      properties:
                        version:
                          type: string
                      type: object
                    kubernetesAPIs:
                      description: 'Optional: Get running KubernetesAPIs
                        from cluster, defaults (empty) to retrieving
                        the APIs from the cluster. Can be manually
                        supplied instead, e.g ["group/version",
                        "group2/version2"]'
                      properties:
                        groupVersions:
                          items:
                            type: string
                          type: array
                      type: object
                    kubernetesVersion:
                      description: 'Optional: Get running Kubernetes
                        version from cluster, defaults (empty)
                        to retrieving the version from the cluster.
                        Can be manually supplied instead.'
                      properties:
                        version:
                          type: string
                      type: object
                    name:
                      type: string
                  type: object
                type: array
            type: object
          path:
            type: string
          secretRef:
            properties:
              name:
                type: string
            type: object
        type: object
      type: array
  type: object
helmTemplate:
  description: Use helm template command to render helm chart
  properties:
    kubernetesAPIs:
      description: 'Optional: Use kubernetes group/versions resources
```

```
    available in the live cluster'
  properties:
    groupVersions:
      items:
        type: string
      type: array
  type: object
kubernetesVersion:
  description: 'Optional: Get Kubernetes version, defaults
    (empty) to retrieving the version from the cluster. Can
    be manually overridden to a value instead.'
  properties:
    version:
      type: string
  type: object
name:
  description: Set name explicitly, default is App CR's name
    (optional; v0.13.0+)
  type: string
namespace:
  description: Set namespace explicitly, default is App CR's
    namespace (optional; v0.13.0+)
  type: string
path:
  description: Path to chart (optional; v0.13.0+)
  type: string
valuesFrom:
  description: One or more secrets, config maps, paths that
    provide values (optional)
  items:
    properties:
      configMapRef:
        properties:
          name:
            type: string
        type: object
      downwardAPI:
        properties:
          items:
            items:
              properties:
                fieldPath:
                  description: 'Required: Selects a field
                    of the app: only annotations, labels,
                    uid, name and namespace are supported.'
                  type: string
                kappControllerVersion:
                  description: 'Optional: Get running KappController
                    version, defaults (empty) to retrieving
                    the current running version.. Can be manually
                    supplied instead.'
                  properties:
                    version:
                      type: string
                  type: object
```

```
                    kubernetesAPIs:
                      description: 'Optional: Get running KubernetesAPIs
                        from cluster, defaults (empty) to retrieving
                        the APIs from the cluster. Can be manually
                        supplied instead, e.g ["group/version",
                        "group2/version2"]'
                      properties:
                        groupVersions:
                          items:
                            type: string
                          type: array
                      type: object
                    kubernetesVersion:
                      description: 'Optional: Get running Kubernetes
                        version from cluster, defaults (empty)
                        to retrieving the version from the cluster.
                        Can be manually supplied instead.'
                      properties:
                        version:
                          type: string
                      type: object
                    name:
                      type: string
                  type: object
                type: array
            type: object
          path:
            type: string
          secretRef:
            properties:
              name:
                type: string
            type: object
        type: object
      type: array
  type: object
jsonnet:
  description: TODO implement jsonnet
  type: object
kbld:
  description: Use kbld to resolve image references to use digests
  properties:
    paths:
      items:
        type: string
      type: array
  type: object
kustomize:
  description: TODO implement kustomize
  type: object
sops:
  description: Use sops to decrypt *.sops.yml files (optional;
    v0.11.0+)
  properties:
    age:
```

```
                        properties:
                          privateKeysSecretRef:
                            description: Secret with private armored PGP private
                              keys (required)
                            properties:
                              name:
                                type: string
                            type: object
                        type: object
                      paths:
                        description: Lists paths to decrypt explicitly (optional;
                          v0.13.0+)
                        items:
                          type: string
                        type: array
                      pgp:
                        description: Use PGP to decrypt files (required)
                        properties:
                          privateKeysSecretRef:
                            description: Secret with private armored PGP private
                              keys (required)
                            properties:
                              name:
                                type: string
                            type: object
                        type: object
                    type: object
                  ytt:
                    description: Use ytt to template configuration
                    properties:
                      fileMarks:
                        description: Control metadata about input files passed to
                          ytt (optional; v0.18.0+) see https://carvel.dev/ytt/docs/latest/
file-marks/

                          for more details
                        items:
                          type: string
                        type: array
                      ignoreUnknownComments:
                        description: Ignores comments that ytt doesn't recognize
                          (optional; default=false)
                        type: boolean
                      inline:
                        description: Specify additional files, including data values
                          (optional)
                        properties:
                          paths:
                            additionalProperties:
                              type: string
                            description: Specifies mapping of paths to their content;
                              not recommended for sensitive values as CR is not
                              encrypted (optional)
                            type: object
                          pathsFrom:
                            description: Specifies content via secrets and config
```

```
                maps; data values are recommended to be placed in
                secrets (optional)
              items:
                properties:
                  configMapRef:
                    properties:
                      directoryPath:
                        description: Specifies where to place files
                          found in secret (optional)
                        type: string
                      name:
                        type: string
                    type: object
                  secretRef:
                    properties:
                      directoryPath:
                        description: Specifies where to place files
                          found in secret (optional)
                        type: string
                      name:
                        type: string
                    type: object
                type: object
              type: array
          type: object
        paths:
          description: Lists paths to provide to ytt explicitly (optional)
          items:
            type: string
          type: array
        strict:
          description: Forces strict mode https://github.com/k14s/ytt/blob/
develop/docs/strict.md
            (optional; default=false)
          type: boolean
        valuesFrom:
          description: Provide values via ytt's --data-values-file
            (optional; v0.19.0-alpha.9)
          items:
            properties:
              configMapRef:
                properties:
                  name:
                    type: string
                type: object
              downwardAPI:
                properties:
                  items:
                    items:
                      properties:
                        fieldPath:
                          description: 'Required: Selects a field
                            of the app: only annotations, labels,
                            uid, name and namespace are supported.'
                          type: string
```

```
                                      kappControllerVersion:
                                        description: 'Optional: Get running KappController
                                          version, defaults (empty) to retrieving
                                          the current running version.. Can be manually
                                          supplied instead.'
                                        properties:
                                          version:
                                            type: string
                                        type: object
                                      kubernetesAPIs:
                                        description: 'Optional: Get running KubernetesAPIs
                                          from cluster, defaults (empty) to retrieving
                                          the APIs from the cluster. Can be manually
                                          supplied instead, e.g ["group/version",
                                          "group2/version2"]'
                                        properties:
                                          groupVersions:
                                            items:
                                              type: string
                                            type: array
                                        type: object
                                      kubernetesVersion:
                                        description: 'Optional: Get running Kubernetes
                                          version from cluster, defaults (empty)
                                          to retrieving the version from the cluster.
                                          Can be manually supplied instead.'
                                        properties:
                                          version:
                                            type: string
                                        type: object
                                      name:
                                        type: string
                                    type: object
                                  type: array
                              type: object
                            path:
                              type: string
                            secretRef:
                              properties:
                                name:
                                  type: string
                              type: object
                          type: object
                        type: array
                    type: object
                type: object
              type: array
          type: object
        status:
          properties:
            conditions:
              items:
                properties:
                  message:
                    description: Human-readable message indicating details about
```

```
          last transition.
        type: string
      reason:
        description: Unique, this should be a short, machine understandable
          string that gives the reason for condition's last transition.
          If it reports "ResizeStarted" that means the underlying persistent
          volume is being resized.
        type: string
      status:
        type: string
      type:
        description: ConditionType represents reconciler state
        type: string
    required:
    - status
    - type
    type: object
  type: array
consecutiveReconcileFailures:
  type: integer
consecutiveReconcileSuccesses:
  type: integer
deploy:
  properties:
    error:
      type: string
    exitCode:
      type: integer
    finished:
      type: boolean
    kapp:
      description: KappDeployStatus contains the associated AppCR deployed
        resources
      properties:
        associatedResources:
          description: AssociatedResources contains the associated App
            label, namespaces and GKs
          properties:
            groupKinds:
              items:
                description: GroupKind specifies a Group and a Kind,
                  but does not force a version.  This is useful for
                  identifying concepts during lookup stages without
                  having partially valid types
                properties:
                  group:
                    type: string
                  kind:
                    type: string
                required:
                - group
                - kind
                type: object
              type: array
            label:
```

```
                      type: string
                    namespaces:
                      items:
                        type: string
                      type: array
                  type: object
              type: object
            startedAt:
              format: date-time
              type: string
            stderr:
              type: string
            stdout:
              type: string
            updatedAt:
              format: date-time
              type: string
          type: object
        fetch:
          properties:
            error:
              type: string
            exitCode:
              type: integer
            startedAt:
              format: date-time
              type: string
            stderr:
              type: string
            stdout:
              type: string
            updatedAt:
              format: date-time
              type: string
          type: object
        friendlyDescription:
          type: string
        inspect:
          properties:
            error:
              type: string
            exitCode:
              type: integer
            stderr:
              type: string
            stdout:
              type: string
            updatedAt:
              format: date-time
              type: string
          type: object
        managedAppName:
          type: string
        observedGeneration:
          description: Populated based on metadata.generation when controller
```

```
                   observes a change to the resource; if this value is out of data,
                   other status fields do not reflect latest state
                 format: int64
                 type: integer
               template:
                 properties:
                   error:
                     type: string
                   exitCode:
                     type: integer
                   stderr:
                     type: string
                   updatedAt:
                     format: date-time
                     type: string
                 type: object
               usefulErrorMessage:
                 type: string
             type: object
         required:
         - spec
         type: object
     served: true
     storage: true
     subresources:
       status: {}
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: packageinstalls.packaging.carvel.dev
spec:
  group: packaging.carvel.dev
  names:
    categories:
    - carvel
    kind: PackageInstall
    listKind: PackageInstallList
    plural: packageinstalls
    shortNames:
    - pkgi
    singular: packageinstall
  scope: Namespaced
  versions:
  - additionalPrinterColumns:
    - description: PackageMetadata name
      jsonPath: .spec.packageRef.refName
      name: Package name
      type: string
    - description: PackageMetadata version
      jsonPath: .status.version
      name: Package version
      type: string
    - description: Friendly description
      jsonPath: .status.friendlyDescription
```

```
      name: Description
      type: string
    - description: Time since creation
      jsonPath: .metadata.creationTimestamp
      name: Age
      type: date
    name: v1alpha1
    schema:
      openAPIV3Schema:
        description: A Package Install is an actual installation of a package and
          its underlying resources on a Kubernetes cluster. It is represented in kapp-
controller
          by a PackageInstall CR. A PackageInstall CR must reference a Package CR.
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
              submits requests to. Cannot be updated. In CamelCase. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            properties:
              canceled:
                description: Canceled when set to true will stop all active changes
                type: boolean
              cluster:
                description: Specifies that Package should be deployed to destination
                  cluster; by default, cluster is same as where this resource resides
                  (optional)
                properties:
                  kubeconfigSecretRef:
                    description: Specifies secret containing kubeconfig (required)
                    properties:
                      key:
                        description: Specifies key that contains kubeconfig (optional)
                        type: string
                      name:
                        description: Specifies secret name within app's namespace
                          (required)
                        type: string
                    type: object
                  namespace:
                    description: Specifies namespace in destination cluster (optional)
                    type: string
                type: object
              noopDelete:
                description: When NoopDelete set to true, PackageInstall deletion
```

```
                should delete PackageInstall/App CR but preserve App's associated
                resources.
              type: boolean
            packageRef:
              description: Specifies the name of the package to install (required)
              properties:
                refName:
                  type: string
                versionSelection:
                  properties:
                    constraints:
                      type: string
                    prereleases:
                      properties:
                        identifiers:
                          items:
                            type: string
                          type: array
                      type: object
                  type: object
              type: object
            paused:
              description: Paused when set to true will ignore all pending changes,
                once it set back to false, pending changes will be applied
              type: boolean
            serviceAccountName:
              description: Specifies service account that will be used to install
                underlying package contents
              type: string
            syncPeriod:
              description: Controls frequency of App reconciliation in time + unit
                format. Always >= 30s. If value below 30s is specified, 30s will
                be used.
              type: string
            values:
              description: Values to be included in package's templating step (currently
                only included in the first templating step) (optional)
              items:
                properties:
                  secretRef:
                    properties:
                      key:
                        type: string
                      name:
                        type: string
                    type: object
                type: object
              type: array
          type: object
        status:
          properties:
            conditions:
              items:
                properties:
                  message:
```

```
                      description: Human-readable message indicating details about
                        last transition.
                      type: string
                    reason:
                      description: Unique, this should be a short, machine understandable
                        string that gives the reason for condition's last transition.
                        If it reports "ResizeStarted" that means the underlying persistent
                        volume is being resized.
                      type: string
                    status:
                      type: string
                    type:
                      description: ConditionType represents reconciler state
                      type: string
                  required:
                  - status
                  - type
                  type: object
                type: array
              friendlyDescription:
                type: string
              lastAttemptedVersion:
                description: LastAttemptedVersion specifies what version was last
                  attempted to be installed. It does _not_ indicate it was successfully
                  installed.
                type: string
              observedGeneration:
                description: Populated based on metadata.generation when controller
                  observes a change to the resource; if this value is out of data,
                  other status fields do not reflect latest state
                format: int64
                type: integer
              usefulErrorMessage:
                type: string
              version:
                description: TODO this is desired resolved version (not actually deployed)
                type: string
            type: object
        required:
        - spec
        type: object
    served: true
    storage: true
    subresources:
      status: {}
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  annotations:
    packaging.carvel.dev/global-namespace: kapp-controller-packaging-global
  name: packagerepositories.packaging.carvel.dev
spec:
  group: packaging.carvel.dev
  names:
```

```
    categories:
    - carvel
    kind: PackageRepository
    listKind: PackageRepositoryList
    plural: packagerepositories
    shortNames:
    - pkgr
    singular: packagerepository
  scope: Namespaced
  versions:
  - additionalPrinterColumns:
    - description: Time since creation
      jsonPath: .metadata.creationTimestamp
      name: Age
      type: date
    - description: Friendly description
      jsonPath: .status.friendlyDescription
      name: Description
      type: string
    name: v1alpha1
    schema:
      openAPIV3Schema:
        description: A package repository is a collection of packages and their metadata.
          Similar to a maven repository or a rpm repository, adding a package repository
          to a cluster gives users of that cluster the ability to install any of the
          packages from that repository.
        properties:
          apiVersion:
            description: 'APIVersion defines the versioned schema of this representation
              of an object. Servers should convert recognized schemas to the latest
              internal value, and may reject unrecognized values. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources'
            type: string
          kind:
            description: 'Kind is a string value representing the REST resource this
              object represents. Servers may infer this from the endpoint the client
              submits requests to. Cannot be updated. In CamelCase. More info: https://
git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds'
            type: string
          metadata:
            type: object
          spec:
            properties:
              fetch:
                properties:
                  git:
                    description: Uses git to clone repository containing package list
                    properties:
                      lfsSkipSmudge:
                        description: Skip lfs download (optional)
                        type: boolean
                      ref:
                        description: Branch, tag, commit; origin is the name of the
                          remote (optional)
                        type: string
```

```
    refSelection:
      description: Specifies a strategy to resolve to an explicit
        ref (optional; v0.24.0+)
      properties:
        semver:
          properties:
            constraints:
              type: string
            prereleases:
              properties:
                identifiers:
                  items:
                    type: string
                  type: array
              type: object
          type: object
      type: object
    secretRef:
      description: 'Secret with auth details. allowed keys: ssh-privatekey,
        ssh-knownhosts, username, password (optional) (if ssh-knownhosts
        is not specified, git will not perform strict host checking)'
      properties:
        name:
          description: Object is expected to be within same namespace
          type: string
      type: object
    subPath:
      description: Grab only portion of repository (optional)
      type: string
    url:
      description: http or ssh urls are supported (required)
      type: string
  type: object
http:
  description: Uses http library to fetch file containing packages
  properties:
    secretRef:
      description: 'Secret to provide auth details (optional) Secret
        may include one or more keys: username, password'
      properties:
        name:
          description: Object is expected to be within same namespace
          type: string
      type: object
    sha256:
      description: Checksum to verify after download (optional)
      type: string
    subPath:
      description: Grab only portion of download (optional)
      type: string
    url:
      description: 'URL can point to one of following formats: text,
        tgz, zip http and https url are supported; plain file, tgz
        and tar types are supported (required)'
      type: string
```

```
                     type: object
                   image:
                     description: Image url; unqualified, tagged, or digest references
                       supported (required)
                     properties:
                       secretRef:
                         description: 'Secret may include one or more keys: username,
                           password, token. By default anonymous access is used for
                           authentication.'
                         properties:
                           name:
                             description: Object is expected to be within same namespace
                             type: string
                         type: object
                       subPath:
                         description: Grab only portion of image (optional)
                         type: string
                       tagSelection:
                         description: Specifies a strategy to choose a tag (optional;
                           v0.24.0+) if specified, do not include a tag in url key
                         properties:
                           semver:
                             properties:
                               constraints:
                                 type: string
                               prereleases:
                                 properties:
                                   identifiers:
                                     items:
                                       type: string
                                     type: array
                                 type: object
                             type: object
                         type: object
                       url:
                         description: 'Docker image url; unqualified, tagged, or digest
                           references supported (required) Example: username/app1-
config:v0.1.0'
                         type: string
                     type: object
                   imgpkgBundle:
                     description: Pulls imgpkg bundle from Docker/OCI registry
                     properties:
                       image:
                         description: Docker image url; unqualified, tagged, or digest
                           references supported (required)
                         type: string
                       secretRef:
                         description: 'Secret may include one or more keys: username,
                           password, token. By default anonymous access is used for
                           authentication.'
                         properties:
                           name:
                             description: Object is expected to be within same namespace
                             type: string
```

```
                type: object
              tagSelection:
                description: Specifies a strategy to choose a tag (optional;
                  v0.24.0+) if specified, do not include a tag in url key
                properties:
                  semver:
                    properties:
                      constraints:
                        type: string
                      prereleases:
                        properties:
                          identifiers:
                            items:
                              type: string
                            type: array
                        type: object
                    type: object
                type: object
            type: object
          inline:
            description: Pull content from within this resource; or other
              resources in the cluster
            properties:
              paths:
                additionalProperties:
                  type: string
                description: Specifies mapping of paths to their content;
                  not recommended for sensitive values as CR is not encrypted
                  (optional)
                type: object
              pathsFrom:
                description: Specifies content via secrets and config maps;
                  data values are recommended to be placed in secrets (optional)
                items:
                  properties:
                    configMapRef:
                      properties:
                        directoryPath:
                          description: Specifies where to place files found
                            in secret (optional)
                          type: string
                        name:
                          type: string
                      type: object
                    secretRef:
                      properties:
                        directoryPath:
                          description: Specifies where to place files found
                            in secret (optional)
                          type: string
                        name:
                          type: string
                      type: object
                  type: object
                type: array
```

```
                type: object
            type: object
          paused:
            description: Paused when set to true will ignore all pending changes,
              once it set back to false, pending changes will be applied
            type: boolean
          syncPeriod:
            description: Controls frequency of PackageRepository reconciliation
            type: string
        required:
        - fetch
        type: object
      status:
        properties:
          conditions:
            items:
              properties:
                message:
                  description: Human-readable message indicating details about
                    last transition.
                  type: string
                reason:
                  description: Unique, this should be a short, machine understandable
                    string that gives the reason for condition's last transition.
                    If it reports "ResizeStarted" that means the underlying persistent
                    volume is being resized.
                  type: string
                status:
                  type: string
                type:
                  description: ConditionType represents reconciler state
                  type: string
              required:
              - status
              - type
              type: object
            type: array
          consecutiveReconcileFailures:
            type: integer
          consecutiveReconcileSuccesses:
            type: integer
          deploy:
            properties:
              error:
                type: string
              exitCode:
                type: integer
              finished:
                type: boolean
              kapp:
                description: KappDeployStatus contains the associated AppCR deployed
                  resources
                properties:
                  associatedResources:
                    description: AssociatedResources contains the associated App
```

```
            label, namespaces and GKs
          properties:
            groupKinds:
              items:
                description: GroupKind specifies a Group and a Kind,
                  but does not force a version.  This is useful for
                  identifying concepts during lookup stages without
                  having partially valid types
                properties:
                  group:
                    type: string
                  kind:
                    type: string
                required:
                - group
                - kind
                type: object
              type: array
            label:
              type: string
            namespaces:
              items:
                type: string
              type: array
          type: object
      type: object
    startedAt:
      format: date-time
      type: string
    stderr:
      type: string
    stdout:
      type: string
    updatedAt:
      format: date-time
      type: string
  type: object
fetch:
  properties:
    error:
      type: string
    exitCode:
      type: integer
    startedAt:
      format: date-time
      type: string
    stderr:
      type: string
    stdout:
      type: string
    updatedAt:
      format: date-time
      type: string
  type: object
friendlyDescription:
```

```
                type: string
              observedGeneration:
                description: Populated based on metadata.generation when controller
                  observes a change to the resource; if this value is out of data,
                  other status fields do not reflect latest state
                format: int64
                type: integer
              template:
                properties:
                  error:
                    type: string
                  exitCode:
                    type: integer
                  stderr:
                    type: string
                  updatedAt:
                    format: date-time
                    type: string
                type: object
              usefulErrorMessage:
                type: string
            type: object
        required:
        - spec
        type: object
    served: true
    storage: true
    subresources:
      status: {}
---
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kapp-controller.carvel.dev/version: v0.45.2
    kbld.k14s.io/images: |
      - origins:
        - local:
            path: /home/runner/work/kapp-controller/kapp-controller
        - git:
            dirty: true
            remoteURL: https://github.com/carvel-dev/kapp-controller
            sha: e3beee23d49899bfc681c9d980c1a3bdc0fa14ac
            tags:
            - v0.45.2
        url: ghcr.io/carvel-dev/kapp-
controller@sha256:d5c5b259d10f8a561fe6717a735ceb053ccb13320f55428977d1d8df46b9bc0d
  name: kapp-controller
  namespace: tkg-system
spec:
  replicas: 1
  revisionHistoryLimit: 0
  selector:
    matchLabels:
      app: kapp-controller
```

```
  template:
    metadata:
      labels:
        app: kapp-controller
    spec:
      containers:
      - args:
        - -packaging-global-namespace=kapp-controller-packaging-global
        - -enable-api-priority-and-fairness=True
        - -tls-cipher-suites=
        env:
        - name: KAPPCTRL_MEM_TMP_DIR
          value: /etc/kappctrl-mem-tmp
        - name: KAPPCTRL_SIDECAREXEC_SOCK
          value: /etc/kappctrl-mem-tmp/sidecarexec.sock
        - name: KAPPCTRL_SYSTEM_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: KAPPCTRL_API_PORT
          value: "10350"
        image: ghcr.io/carvel-dev/kapp-
controller@sha256:d5c5b259d10f8a561fe6717a735ceb053ccb13320f55428977d1d8df46b9bc0d
        name: kapp-controller
        ports:
        - containerPort: 10350
          name: api
          protocol: TCP
        resources:
          requests:
            cpu: 120m
            memory: 100Mi
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
          readOnlyRootFilesystem: true
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
        volumeMounts:
        - mountPath: /etc/kappctrl-mem-tmp
          name: template-fs
        - mountPath: /home/kapp-controller
          name: home
      - args:
        - --sidecarexec
        env:
        - name: KAPPCTRL_SIDECAREXEC_SOCK
          value: /etc/kappctrl-mem-tmp/sidecarexec.sock
        - name: IMGPKG_ACTIVE_KEYCHAINS
          value: gke,aks,ecr
        image: ghcr.io/carvel-dev/kapp-
controller@sha256:d5c5b259d10f8a561fe6717a735ceb053ccb13320f55428977d1d8df46b9bc0d
```

```
        name: kapp-controller-sidecarexec
        resources:
          requests:
            cpu: 120m
            memory: 100Mi
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
          readOnlyRootFilesystem: false
          runAsNonRoot: true
          seccompProfile:
            type: RuntimeDefault
        volumeMounts:
        - mountPath: /etc/kappctrl-mem-tmp
          name: template-fs
        - mountPath: /home/kapp-controller
          name: home
        - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
          name: empty-sa
      serviceAccount: kapp-controller-sa
      volumes:
      - emptyDir:
          medium: Memory
        name: template-fs
      - emptyDir:
          medium: Memory
        name: home
      - emptyDir: {}
        name: empty-sa
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kapp-controller-sa
  namespace: tkg-system
```

# Instalar el administrador de certificados en TKr para vSphere 7.x

Consulte estas instrucciones para instalar el administrador de certificados en un TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar administrador de certificados

Instale el administrador de certificados.

1 Indique las versiones de paquetes disponibles del administrador de certificados.

```
kubectl -n tkg-system get packages | grep cert-manager
```

2 Cree `cert-manager.yaml` con la versión de destino.

   Consulte cert-manager.yaml.

3 Instale el administrador de certificados.

```
kubectl apply -f cert-manager.yaml
```

   Resultado esperado:

```
serviceaccount/cert-manager-sa created
clusterrolebinding.rbac.authorization.k8s.io/admin created
packageinstall.packaging.carvel.dev/cert-manager created
secret/cert-manager-data-values created
```

4 Compruebe la instalación del administrador de certificados.

```
kubectl get pkgi -A
```

   Resultado esperado:

```
NAMESPACE    NAME           PACKAGE NAME                     PACKAGE VERSION
DESCRIPTION           AGE
tkg-system   cert-manager   cert-manager.tanzu.vmware.com   1.12.2+vmware.2-
tkg.2   Reconcile succeeded   57s
```

5 Compruebe los pods del administrador de certificados.

```
kubectl get pods -A
```

```
NAMESPACE           NAME                                          READY   STATUS
RESTARTS      AGE
tkg-system        cert-manager-666586c866-826rz                 1/1      Running
0             48s
tkg-system        cert-manager-cainjector-68697ccc4b-xbfff     1/1      Running
0             48s
tkg-system        cert-manager-webhook-57ccbd4db9-tzw4c        1/1      Running
0             48s
```

## cert-manager.yaml

Consulte el siguiente ejemplo `cert-manager.yaml` para instalar el administrador de certificados. Actualice la variable de la versión para que coincida con la versión del paquete de destino.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cert-manager-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```

```
  name: admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: cert-manager-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: cert-manager
  namespace: tkg-system
spec:
  serviceAccountName: cert-manager-sa
  packageRef:
    refName: cert-manager.tanzu.vmware.com
    versionSelection:
      constraints: 1.12.2+vmware.2-tkg.2 #PKG-VERSION
  values:
  - secretRef:
      name: cert-manager-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: cert-manager-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    namespace: tkg-system
```

# Instalar Contour en TKr para vSphere 7.x

Consulte estas instrucciones para instalar paquetes estándar en un clúster de TKG aprovisionado con un TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar Contour con Envoy

Instale la entrada de Contour con el servicio Envoy.

1 Enumere las versiones de Contour disponibles existentes en el repositorio.

```
kubectl get packages -n tkg-system | grep contour
```

2 Cree la especificación `contour.yaml`.

Consulte #unique_187/unique_187_Connect_42_GUID-CC995CF8-0F4B-4D92-A782-A3832C0EA5AE.

3 Si es necesario, personalice `contour-data-values` en función de su entorno.

   Consulte Referencia del paquete de Contour.

4 Instale Contour.

```
kubectl apply -f contour.yaml
```

```
serviceaccount/contour-sa
createdclusterrolebinding.rbac.authorization.k8s.io/contour-role-binding created
packageinstall.packaging.carvel.dev/contour created
secret/contour-data-values created
```

5 Compruebe la instalación del paquete de Contour.

```
kubectl get pkgi -A
```

6 Compruebe los objetos de Contour.

```
kubectl get all -n contour-ingress
```

```
NAME                            READY   STATUS    RESTARTS   AGE
pod/contour-777bdddc69-fqnsp    1/1     Running   0          102s
pod/contour-777bdddc69-gs5xv    1/1     Running   0          102s
pod/envoy-d4jtt                 2/2     Running   0          102s
pod/envoy-g5h72                 2/2     Running   0          102s
pod/envoy-pjpzc                 2/2     Running   0          102s

NAME                TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                        AGE
service/contour     ClusterIP      10.105.242.46   <none>
8001/TCP                       102s
service/envoy       LoadBalancer   10.103.245.57   10.197.154.69   80:32642/
TCP,443:30297/TCP   102s

NAME                     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/envoy     3         3         3       3            3
<none>          102s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/contour    2/2     2            2           102s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/contour-777bdddc69    2         2         2       102s
```

El paquete de Contour instala 2 pods Contour y 3 pods Envoy. Tanto Contour como Envoy se exponen como servicios. En este ejemplo, el servicio Envoy tiene una dirección IP externa 10.197.154.69. Esta dirección IP está separada del CIDR especificado para **Red de carga de trabajo > Ingreso**. Se creará una nueva instancia del equilibrador de carga para esta dirección IP. Los miembros del grupo de servidores de este equilibrador de carga son los pods Envoy. Los pods de Envoy asumen las direcciones IP de los nodos de trabajo en los que se ejecutan. Para ver estas direcciones IP, consulte los nodos del clúster (`kubectl get nodes -o wide`).

## contour.yaml

Use el siguiente `contour.yaml` para instalar Contour con Envoy. Actualice la variable de la versión para que coincida con la versión del paquete de destino.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: contour-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: contour-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: contour-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: contour
  namespace: tkg-system
spec:
  serviceAccountName: contour-sa
  packageRef:
    refName: contour.tanzu.vmware.com
    versionSelection:
      constraints: 1.26.1+vmware.1-tkg.1 #PKG-VERSION
  values:
  - secretRef:
      name: contour-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: contour-data-values
  namespace: tkg-system
stringData:
```

```
  values.yml: |
    ---
    namespace: tanzu-system-ingress
    contour:
      configFileContents: {}
      useProxyProtocol: false
      replicas: 2
      pspNames: "vmware-system-restricted"
      logLevel: info
    envoy:
      service:
        type: LoadBalancer
        annotations: {}
        externalTrafficPolicy: Cluster
        disableWait: false
      hostPorts:
        enable: true
        http: 80
        https: 443
      hostNetwork: false
      terminationGracePeriodSeconds: 300
      logLevel: info
    certificates:
      duration: 8760h
      renewBefore: 360h
```

# Instalar ExternalDNS en TKr para vSphere 7.x

Consulte estas instrucciones para instalar ExternalDNS en el clúster de TKG aprovisionado con un TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar ExternalDNS

Instale ExternalDNS en un clúster de TKG aprovisionado con TKr para vSphere 7.x.

1 Enumere las versiones de ExternalDNS disponibles en el repositorio.

```
kubectl get packages -n tkg-system | grep external-dns
```

2 Cree el espacio de nombres de ExternalDNS.

```
kubectl create namespace tanzu-system-service-discovery --dry-run=client -o yaml | kubectl
apply -f -
```

3 Establezca la posición de seguridad en el espacio de nombres.

```
kubectl label namespace tanzu-system-service-discovery pod-security.kubernetes.io/
enforce=privileged
```

4 Prepare el YAML de implementación de enlace.

Consulte bind-deployment.yaml.

5 Implemente el servidor DNS de enlace.

```
kubectl apply -n tanzu-system-service-discovery -f bind-deployment.yaml
```

6 Prepare el YAML de implementación de ExternalDNS.

Consulte external-dns-deploy.yaml.

7 Cree un secreto con el archivo `external-dns-default-values.yaml`.

```
svcip=$(kubectl get svc bind -n tanzu-system-service-discovery -o
jsonpath='{.spec.clusterIP}')sed -i "s/--rfc2136-host=[0-9.]\+/--rfc2136-host=$svcip/g"
external-dns-deploy.yaml
```

```
kubectl create secret generic external-dns-default-values --from-file=values.yaml=external-
dns-deploy.yaml -n tkg-system
```

8 Compruebe el secreto.

```
kubectl get secret external-dns-default-values -n tkg-system
```

```
kubectl get secret external-dns-default-values -n tkg-system -oyaml
```

9 Prepare el YAML de instalación de paquetes de ExternalDNS.

Consulte external-dns-packageinstall.yaml.

10 Configure el enlace.

```
sed -i "s/--rfc2136-host=[0-9.]\+/--rfc2136-host=$svcip/g" external-dns-packageinstall.yaml
```

11 Cree el paquete de DNS externo.

```
kubectl apply -f external-dns-packageinstall.yaml
```

12 Compruebe la instalación de ExternalDNS.

```
kubectl get all -n tanzu-system-service-discovery
```

## bind-deployment.yaml

Ejemplo `bind-deployment.yaml`.

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: bind-config
data:
  named.conf: |
    key "externaldns-key" {
```

```
            algorithm hmac-sha256;
            secret "O0DhTJzZ0GjfuQmB9TBc1ELchv5oDMTlQs3NNOdMZJU=";
    };

    # bind needs to recurse to coredns in the case of resolving CNAME records
    # it may know about to A records. E.g This test runs on AWS which uses
    # CNAMEs for their LoadBalancer Services and bind will want to resolve
    # those CNAME records to A records using an upstream DNS server.
    options {
        recursion yes;
        forwarders {
            COREDNS_CLUSTER_IP;
        };
        forward only;
        dnssec-enable yes;
        dnssec-validation yes;
    };

    zone "k8s.example.org" {
        type master;
        file "/etc/bind/k8s.zone";
        allow-transfer {
            key "externaldns-key";
        };
        update-policy {
            grant externaldns-key zonesub ANY;
        };
    };
  k8s.zone: |
    $TTL 60 ; 1 minute
    @         IN SOA  k8s.example.org. root.k8s.example.org. (
                                  16         ; serial
                                  60         ; refresh (1 minute)
                                  60         ; retry (1 minute)
                                  60         ; expire (1 minute)
                                  60         ; minimum (1 minute)
                                  )
                          NS      ns.k8s.example.org.
    ns                    A       1.2.3.4
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bind
spec:
  selector:
    matchLabels:
      app: bind
  template:
    metadata:
      labels:
        app: bind
    spec:
      containers:
      - name: bind
```

```
        image: docker.io/internetsystemsconsortium/bind9:9.16
        imagePullPolicy: IfNotPresent
        command:
        - 'sh'
        - '-c'
        - |
          /usr/sbin/named -g -c /etc/bind/named.conf
        ports:
        - containerPort: 53
          name: dns
          protocol: UDP
        - containerPort: 53
          name: dns-tcp
          protocol: TCP
        volumeMounts:
          - name: named-conf-volume
            mountPath: /etc/bind/named.conf
            subPath: named.conf
          - name: k8s-zone-volume
            mountPath: /etc/bind/k8s.zone
            subPath: k8s.zone
      volumes:
        - name: data
          emptyDir: {}
        - name: named-conf-volume
          configMap:
            name: bind-config
            items:
            - key: named.conf
              path: named.conf
        - name: k8s-zone-volume
          configMap:
            name: bind-config
            items:
            - key: k8s.zone
              path: k8s.zone
---
apiVersion: v1
kind: Service
metadata:
  name: bind
  labels:
    app: bind
spec:
  selector:
    app: bind
  type: ClusterIP
  ports:
  - port: 53
    targetPort: 53
    protocol: TCP
    name: dns-tcp
```

```
  - port: 53
    targetPort: 53
    protocol: UDP
    name: dns
```

## external-dns-deploy.yaml

Ejemplo `external-dns-deploy.yaml`.

```
deployment:
  args:
  - --source=service
  - --source=ingress
  - --txt-owner-id=k8s
  - --domain-filter=k8s.example.org
  - --namespace=default
  - --provider=rfc2136
  - --rfc2136-host=198.201.49.227
  - --rfc2136-port=53
  - --rfc2136-zone=k8s.example.org
  - --rfc2136-tsig-secret=O0DhTJzZ0GjfuQmB9TBc1ELchv5oDMTlQs3NNOdMZJU=
  - --rfc2136-tsig-secret-alg=hmac-sha256
  - --rfc2136-tsig-keyname=externaldns-key
```

## external-dns-packageinstall.yaml

El siguiente ejemplo se puede utilizar para BIND. Actualice la versión del paquete según sea necesario.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns-default-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dns-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: external-dns-default-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: dns
  namespace: tkg-system
spec:
```

```
  serviceAccountName: external-dns-default-sa
  packageRef:
    refName: external-dns.tanzu.vmware.com
    versionSelection:
      constraints: 0.13.6+vmware.1-tkg.1
  values:
  - secretRef:
      name: external-dns-default-values
---
apiVersion: v1
kind: Secret
metadata:
  name: external-dns-reg-creds
  namespace: tanzu-system-service-discovery
stringData:
  values.yml: |
    ---
    namespace: tanzu-system-service-discovery
    dns:
      deployment:
        args:
        - --txt-owner-id=k8s
        - --provider=rfc2136
        - --rfc2136-host=198.201.49.227 #! IP of compatible DNS server
        - --rfc2136-port=53
        - --rfc2136-zone=mk8s.example.org #! zone where services are deployed
        - --rfc2136-tsig-secret=O0DhTJzZ0GjfuQmB9TBc1ELchv5oDMTlQs3NNOdMZJU= #! TSIG secret
authorized to update DNS
        - --rfc2136-tsig-secret-alg=hmac-sha256
        - --rfc2136-tsig-keyname=externaldns-key
        - --rfc2136-tsig-axfr
        - --source=service
        - --source=ingress
        - --domain-filter=k8s.example.org1 #! zone where services are deployed
```

El siguiente ejemplo se puede utilizar para el proveedor de DNS de AWS (ruta 53). Actualice la versión del paquete según sea necesario.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dns-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dns-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: dns-sa
```

```
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: dns
  namespace: tkg-system
spec:
  serviceAccountName: dns-sa
  packageRef:
    refName: dns.tanzu.vmware.com
    versionSelection:
      constraints: 0.13.6+vmware.1-tkg.1
  values:
  - secretRef:
      name: dns-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: dns-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    namespace: tanzu-system-service-discovery
    dns:
      pspNames: "vmware-system-restricted"
      deployment:
        args:
        - --source=service
        - --source=ingress
        - --source=contour-httpproxy #! configure external-dns to read Contour HTTPProxy 
resources
        - --domain-filter=my-zone.example.org #! zone where services are deployed
        - --provider=aws
        - --policy=upsert-only #! prevent deleting any records, omit to enable full sync
        - --aws-zone-type=public #! only look at public hosted zones (public, private, no 
value for both)
        - --aws-prefer-cname
        - --registry=txt
        - --txt-owner-id=ROUTE_53_HOSTED_ZONE_ID #! Route53 hosted zone identifier for my-
zone.example.org
        - --txt-prefix=txt #! disambiguates TXT records from CNAME records
        env:
          - name: AWS_ACCESS_KEY_ID
            valueFrom:
              secretKeyRef:
                name: route53-credentials #! Kubernetes secret for route53 credentials
                key: aws_access_key_id
          - name: AWS_SECRET_ACCESS_KEY
            valueFrom:
              secretKeyRef:
                name: route53-credentials #! Kubernetes secret for route53 credentials
                key: aws_secret_access_key
```

El siguiente ejemplo se puede utilizar para un proveedor de DNS de Azure. Actualice la versión del paquete según sea necesario.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dns-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dns-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: dns-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: dns
  namespace: tkg-system
spec:
  serviceAccountName: dns-sa
  packageRef:
    refName: dns.tanzu.vmware.com
    versionSelection:
      constraints: 0.13.6+vmware.1-tkg.1
  values:
  - secretRef:
      name: dns-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: dns-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    namespace: tanzu-system-service-discovery
    dns:
      pspNames: "vmware-system-restricted"
      deployment:
        args:
        - --provider=azure
        - --source=service
        - --source=ingress
        - --source=contour-httpproxy #! read Contour HTTPProxy resources
        - --domain-filter=my-zone.example.org #! zone where services are deployed
```

```
        - --azure-resource-group=my-resource-group #! Azure resource group
        volumeMounts:
        - name: azure-config-file
          mountPath: /etc/kubernetes
          readOnly: true
        #@overlay/replace
        volumes:
        - name: azure-config-file
          secret:
            secretName: azure-config-file
```

# Instalar Fluent Bit en TKr para vSphere 7.x

Consulte estas instrucciones para instalar Fluent Bit en un clúster de TKG aprovisionado con TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar Fluent Bit

Instale Fluent Bit para el reenvío de registros.

1 Enumere las versiones de Fluent Bit disponibles en el repositorio.

```
kubectl -n tkg-system get packages | grep fluent-bit
```

2 Cree el espacio de nombres.

```
kubectl create ns tanzu-system-logging
```

3 Etiquete el espacio de nombres para PSA.

```
kubectl label ns fluentbit-logging pod-security.kubernetes.io/enforce=privileged
```

O, alternativamente:

```
apiVersion: v1
kind: Namespace
metadata:
  name: fluentbit-logging
---
apiVersion: v1
kind: Namespace
metadata:
  name: fluentbit-logging
  labels: pod-security.kubernetes.io/enforce: privileged
```

4 Prepare `fluentbit.yaml`.

Consulte

5 Personalice `fluentbit-data-values` según sea necesario para su entorno.

Consulte Referencia del paquete de Fluent Bit para ver los parámetros de configuración.

6 Instale Fluent Bit.

```
kubectl apply -f fluentbit.yaml
```

7 Compruebe la instalación de Fluent Bit.

```
kubectl get all -n fluentbit-logging
```

## fluentbit.yaml

El siguiente ejemplo se puede utilizar para un endpoint de Syslog. Actualice la versión del paquete según sea necesario.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentbit-sa
  namespace: tkg-system
  annotations:
    pod-security.kubernetes.io/enforce: "privileged"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: fluentbit-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: fluentbit-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: fluentbit
  namespace: tkg-system
spec:
  serviceAccountName: fluentbit-sa
  packageRef:
    refName: fluent-bit.tanzu.vmware.com
    versionSelection:
      constraints: 2.1.6+vmware.1-tkg.2 #PKG_VERSION
  values:
  - secretRef:
      name: fluentbit-data-values
---
apiVersion: v1
kind: Secret
metadata:
```

```
  name: fluentbit-data-values
  namespace: tkg-system
stringData:
  values.yml: |
    ---
    namespace: tanzu-system-logging
    tkg:
      instance_name: "guest-cluster"     #TKG_INSTANCE_NAME
      cluster_name: "tkgs-vc-wl"         #TKG_CLUSTER_NAME
    fluentbit:
      output_plugin: "syslog"
      syslog:
        host: "10.202.27.235"           #SYSLOG_HOST
        port: "514"                     #SYSLOG_PORT
        mode: "tcp"                     #SYSLOG_MODE
        format: "rfc5424"               #SYSLOG_FORMAT
```

# Instalar Prometheus en TKr para vSphere 7.x

Consulte estas instrucciones para instalar Prometheus en un clúster de TKG aprovisionado con TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar Prometheus

Instale Prometheus con Alertmanager.

1 Enumere las versiones de paquetes de Prometheus disponibles en el repositorio.

```
kubectl get packages -n tkg-system | grep prometheus
```

2 Cree el espacio de nombres de Prometheus.

```
kubectl create ns tanzu-system-monitoring
```

3 Configure PSA en el espacio de nombres de Prometheus.

```
kubectl label ns prometheus-monitoring pod-security.kubernetes.io/enforce=privileged
```

```
kubectl get ns prometheus-monitoring -oyaml|grep privileged
```

4 Cree el archivo `prometheus-data-values.yaml`.

Consulte .

5 Cree un secreto con `prometheus-data-values.yaml` como entrada.

**Nota** Debido a que `prometheus-data-values` es grande, es menos propenso a errores crear el secreto por separado en lugar de intentar incluirlo en la especificación de YAML de Prometheus.

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-data-values.yaml -n tkg-system
```

```
secret/prometheus-data-values created
```

6 Compruebe el secreto.

```
kubectl get secrets -A
```

```
kubectl describe secret prometheus-data-values -n tkg-system
```

7 Si es necesario, personalice `prometheus-data-values` en función de su entorno.

Consulte Configuración de Prometheus.

Si actualiza `prometheus-data-values.yaml`, reemplace el secreto con este comando.

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-data-values.yaml -n tkg-system -o yaml --dry-run=client | kubectl replace -f-
```

```
secret/prometheus-data-values replaced
```

8 Cree la especificación `prometheus.yaml`.

Consulte Instalar Prometheus en TKr para vSphere 7.x.

9 Instale Prometheus.

```
kubectl apply -f prometheus.yaml
```

```
serviceaccount/prometheus-sa created
clusterrolebinding.rbac.authorization.k8s.io/prometheus-role-binding created
packageinstall.packaging.carvel.dev/prometheus created
```

10 Compruebe la instalación del paquete de Prometheus.

```
kubectl get pkgi -A
```

11 Compruebe los objetos de Prometheus.

```
kubectl get all -n  tanzu-system-monitoring
```

```
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/alertmanager-757ffd8c6c-97kqd                       1/1     Running   0          87s
pod/prometheus-kube-state-metrics-67b965c5d8-8mf4k      1/1     Running   0          87s
pod/prometheus-node-exporter-4spk9                      1/1     Running   0          87s
pod/prometheus-node-exporter-6k2rh                      1/1     Running   0          87s
pod/prometheus-node-exporter-7z9s8                      1/1     Running   0          87s
pod/prometheus-node-exporter-9d6ss                      1/1     Running   0          87s
pod/prometheus-node-exporter-csbwc                      1/1     Running   0          87s
pod/prometheus-node-exporter-qdb72                      1/1     Running   0          87s
pod/prometheus-pushgateway-dff459565-wfrz5              1/1     Running   0          86s
pod/prometheus-server-56c68567f-bjcn5                   2/2     Running   0          87s

NAME                                    TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)         AGE
service/alertmanager                    ClusterIP   10.109.54.17     <none>
80/TCP          88s
service/prometheus-kube-state-metrics   ClusterIP   None             <none>        80/
TCP,81/TCP   88s
service/prometheus-node-exporter        ClusterIP   10.104.132.133   <none>
9100/TCP        88s
service/prometheus-pushgateway          ClusterIP   10.109.80.171    <none>
9091/TCP        88s
service/prometheus-server               ClusterIP   10.103.252.220   <none>
80/TCP          87s

NAME                                      DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/prometheus-node-exporter   6         6         6       6
6           <none>          88s

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/alertmanager                    1/1     1            1           88s
deployment.apps/prometheus-kube-state-metrics   1/1     1            1           88s
deployment.apps/prometheus-pushgateway          1/1     1            1           87s
deployment.apps/prometheus-server               1/1     1            1           88s

NAME                                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/alertmanager-757ffd8c6c                     1         1         1       88s
replicaset.apps/prometheus-kube-state-metrics-67b965c5d8    1         1         1       88s
replicaset.apps/prometheus-pushgateway-dff459565            1         1         1       87s
replicaset.apps/prometheus-server-56c68567f                 1         1         1       88s
```

12 Compruebe el PVC de Prometheus.

```
kubectl get pvc -n tanzu-system-monitoring
```

```

NAME                STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS             AGE
alertmanager        Bound    pvc-5781956b-abc4-4646-b54c-a3eda1bf140c   2Gi
RWO            vsphere-default-policy   53m
prometheus-server   Bound    pvc-9d45d7cb-6754-40a6-a4b6-f47cf6c949a9   20Gi
RWO            vsphere-default-policy   53m
```

## Acceder al panel de control de Prometheus

Una vez que Prometheus esté instalado, complete los siguientes pasos para acceder al panel de control de Prometheus.

1 Asegúrese de que la sección `ingress` del archivo `prometheus-data-values.yaml` se rellena con todos los campos obligatorios.

```
ingress:
  enabled: true
  virtual_host_fqdn: "prometheus.system.tanzu"
  prometheus_prefix: "/"
  alertmanager_prefix: "/alertmanager/"
  prometheusServicePort: 80
  alertmanagerServicePort: 80
  #! [Optional] The certificate for the ingress if you want to use your own TLS
certificate.
  #! We will issue the certificate by cert-manager when it's empty.
  tlsCertificate:
    #! [Required] the certificate
    tls.crt:
    #! [Required] the private key
    tls.key:
    #! [Optional] the CA certificate
    ca.crt:
```

2 Obtenga la dirección IP pública (externa) del equilibrador de carga de Contour con Envoy.

```
kubectl -n tanzu-system-ingress get all
```

3 Inicie la interfaz web de Prometheus.

```
kubectl get httpproxy -n tanzu-system-monitoring
```

El FQDN debe estar disponible en la dirección IP pública para el servicio Envoy.

```
NAME                    FQDN                        TLS SECRET       STATUS   STATUS
DESCRIPTION
prometheus-httpproxy    prometheus.system.tanzu     prometheus-tls   valid    Valid  HTTPProxy
```

4 Cree un registro de DNS que asigne el FQDN de Prometheus a la dirección IP externa del equilibrador de carga de Envoy.

5 Para acceder al panel de control de Prometheus, desplácese hasta el FQDN de Prometheus con un navegador.

## prometheus-data-values.yaml

```
alertmanager:
  config:
    alertmanager_yml: "global: {}\nreceivers:\n- name: default-receiver\ntemplates:\n\
      - '/etc/alertmanager/templates/*.tmpl'\nroute:\n  group_interval: 5m\n  group_wait:\
      \ 10s\n  receiver: default-receiver\n  repeat_interval: 3h\n"
  deployment:
    containers:
      resources: {}
    podAnnotations: {}
    podLabels: {}
    replicas: 1
    rollingUpdate:
      maxSurge: null
      maxUnavailable: null
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    annotations: {}
    storage: 2Gi
    storageClassName: wcpglobalstorageprofile
  service:
    annotations: {}
    labels: {}
    port: 80
    targetPort: 9093
    type: ClusterIP
ingress:
  alertmanagerServicePort: 80
  alertmanager_prefix: /alertmanager/
  enabled: false
  prometheusServicePort: 80
  prometheus_prefix: /
  tlsCertificate:
    ca.crt: null
    tls.crt: null
    tls.key: null
  virtual_host_fqdn: prometheus.system.tanzu
kube_state_metrics:
  deployment:
    containers:
      resources: {}
    podAnnotations: {}
    podLabels: {}
    replicas: 1
  service:
    annotations: {}
```

```
    labels: {}
    port: 80
    targetPort: 8080
    telemetryPort: 81
    telemetryTargetPort: 8081
    type: ClusterIP
namespace: tanzu-system-monitoring
node_exporter:
  daemonset:
    containers:
      resources: {}
    hostNetwork: false
    podAnnotations: {}
    podLabels: {}
    updatestrategy: RollingUpdate
  service:
    annotations: {}
    labels: {}
    port: 9100
    targetPort: 9100
    type: ClusterIP
prometheus:
  config:
    alerting_rules_yml: '{}

      '
    alerts_yml: '{}

      '
    prometheus_yml: "global:\n  evaluation_interval: 1m\n  scrape_interval: 1m\n \
      \ scrape_timeout: 10s\nrule_files:\n- /etc/config/alerting_rules.yml\n- /etc/config/
recording_rules.yml\n\
      - /etc/config/alerts\n- /etc/config/rules\nscrape_configs:\n- job_name: 'prometheus'\n\
      \  scrape_interval: 5s\n  static_configs:\n  - targets: ['localhost:9090']\n\
      - job_name: 'kube-state-metrics'\n  static_configs:\n  - targets: ['prometheus-kube-
state-metrics.tanzu-system-monitoring.svc.cluster.local:8080']\n\
      \n- job_name: 'node-exporter'\n  static_configs:\n  - targets: ['prometheus-node-
exporter.tanzu-system-monitoring.svc.cluster.local:9100']\n\
      \n- job_name: 'kubernetes-pods'\n  kubernetes_sd_configs:\n  - role: pod\n \
      \ relabel_configs:\n  - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_scrape]\n\
      \    action: keep\n    regex: true\n  - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_path]\n\
      \    action: replace\n    target_label: __metrics_path__\n    regex: (.+)\n\
      \  - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
\n\
      \    action: replace\n    regex: ([^:]+)(?::\\d+)?;(\\d+)\n    replacement:\
      \ $1:$2\n    target_label: __address__\n  - action: labelmap\n    regex:
__meta_kubernetes_pod_label_(.+)\n\
      \  - source_labels: [__meta_kubernetes_namespace]\n    action: replace\n   \
      \ target_label: kubernetes_namespace\n  - source_labels: [__meta_kubernetes_pod_name]\n\
      \    action: replace\n    target_label: kubernetes_pod_name\n- job_name: kubernetes-
nodes-cadvisor\n\
      \  kubernetes_sd_configs:\n  - role: node\n  relabel_configs:\n  - action: labelmap\n\
      \    regex: __meta_kubernetes_node_label_(.+)\n  - replacement:
```

```
kubernetes.default.svc:443\n\
      \    target_label: __address__\n  - regex: (.+)\n    replacement: /api/v1/nodes/$1/
proxy/metrics/cadvisor\n\
      \    source_labels:\n    - __meta_kubernetes_node_name\n    target_label:
__metrics_path__\n\
      \  scheme: https\n  tls_config:\n    ca_file: /var/run/secrets/kubernetes.io/
serviceaccount/ca.crt\n\
      \    insecure_skip_verify: true\n  bearer_token_file: /var/run/secrets/kubernetes.io/
serviceaccount/token\n\
      - job_name: kubernetes-apiservers\n  kubernetes_sd_configs:\n  - role: endpoints\n\
      \  relabel_configs:\n  - action: keep\n    regex: default;kubernetes;https\n\
      \    source_labels:\n    - __meta_kubernetes_namespace\n    -
__meta_kubernetes_service_name\n\
      \    - __meta_kubernetes_endpoint_port_name\n  scheme: https\n  tls_config:\n\
      \    ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt\n
insecure_skip_verify:\
      \ true\n  bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token\n\
      alerting:\n  alertmanagers:\n  - scheme: http\n    static_configs:\n    - targets:\n\
      \      - alertmanager.tanzu-system-monitoring.svc:80\n  - kubernetes_sd_configs:\n\
      \      - role: pod\n    relabel_configs:\n    - source_labels:
[__meta_kubernetes_namespace]\n\
      \      regex: default\n      action: keep\n    - source_labels:
[__meta_kubernetes_pod_label_app]\n\
      \      regex: prometheus\n      action: keep\n    - source_labels:
[__meta_kubernetes_pod_label_component]\n\
      \      regex: alertmanager\n      action: keep\n    - source_labels:
[__meta_kubernetes_pod_annotation_prometheus_io_probe]\n\
      \      regex: .*\n      action: keep\n    - source_labels:
[__meta_kubernetes_pod_container_port_number]\n\
      \      regex:\n      action: drop\n"
    recording_rules_yml: "groups:\n  - name: kube-apiserver.rules\n    interval: 3m\n\
      \    rules:\n    - expr: |2\n        (\n          (\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[1d]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[1d]))\n\
      \              or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[1d]))\n\
      \             +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[1d]))\n\
      \            )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[1d]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[1d]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate1d\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[1h]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[1h]))\n\
```

```
      \                 or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[1h]))\n\
      \               +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[1h]))\n\
      \             )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[1h]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[1h]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate1h\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[2h]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[2h]))\n\
      \                 or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[2h]))\n\
      \               +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[2h]))\n\
      \             )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[2h]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[2h]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate2h\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[30m]))\n            -\n        \
      \    (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[30m]))\n\
      \                 or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[30m]))\n\
      \               +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[30m]))\n\
      \             )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[30m]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[30m]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate30m\n    - expr: |2\n        (\n          (\n  \
      \          # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[3d]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[3d]))\n\
      \                 or\n                vector(0)\n              )\n          \
```

```
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[3d]))\n\
      \              +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[3d]))\n\
      \            )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[3d]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[3d]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate3d\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[5m]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[5m]))\n\
      \                or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[5m]))\n\
      \              +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[5m]))\n\
      \            )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[5m]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[5m]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate5m\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[6h]))\n            -\n         \
      \   (\n              (\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[6h]))\n\
      \                or\n                vector(0)\n              )\n          \
      \    +\n              sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[6h]))\n\
      \              +\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[6h]))\n\
      \            )\n          )\n          +\n          # errors\n
sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",code=~\"5..\"}[6h]))\n        )\n\
      \        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=~\"LIST|GET\"}[6h]))\n      labels:\n        verb: read\n      record:\
      \ apiserver_request:burnrate6h\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\"}[1d]))\n           \
      \ -\n            sum(rate(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[1d]))\n          )\n          +\n\
      \          sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\",code=~\"5..\"}[1d]))\n        )\n        /\n       \
```

```
      \ sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|
DELETE\"\
      }[1d]))\n      labels:\n        verb: write\n      record:
apiserver_request:burnrate1d\n\
      \    - expr: |2\n        (\n          (\n            # too slow\n          \
      \  sum(rate(apiserver_request_duration_seconds_count{job=\"kubernetes-apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\"}[1h]))\n            -\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[1h]))\n  \
      \        )\n          +\n          sum(rate(apiserver_request_total{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",code=~\"5..\"}[1h]))\n        )\n        /\n\
      \        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\"}[1h]))\n      labels:\n        verb: write\n      record:\
      \ apiserver_request:burnrate1h\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\"}[2h]))\n           \
      \ -\n            sum(rate(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[2h]))\n          )\n          +\n\
      \          sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\",code=~\"5..\"}[2h]))\n        )\n        /\n       \
      \ sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|
DELETE\"\
      }[2h]))\n      labels:\n        verb: write\n      record:
apiserver_request:burnrate2h\n\
      \    - expr: |2\n        (\n          (\n            # too slow\n          \
      \  sum(rate(apiserver_request_duration_seconds_count{job=\"kubernetes-apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\"}[30m]))\n            -\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[30m]))\n \
      \         )\n          +\n          sum(rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\",code=~\"5..\"}[30m]))\n\
      \        )\n        /\n        sum(rate(apiserver_request_total{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\"}[30m]))\n      labels:\n        verb: write\n\
      \      record: apiserver_request:burnrate30m\n    - expr: |2\n        (\n  \
      \        (\n            # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\"}[3d]))\n           \
      \ -\n            sum(rate(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[3d]))\n          )\n          +\n\
      \          sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\",code=~\"5..\"}[3d]))\n        )\n        /\n       \
      \ sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|
DELETE\"\
      }[3d]))\n      labels:\n        verb: write\n      record:
apiserver_request:burnrate3d\n\
      \    - expr: |2\n        (\n          (\n            # too slow\n          \
      \  sum(rate(apiserver_request_duration_seconds_count{job=\"kubernetes-apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\"}[5m]))\n            -\n
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[5m]))\n  \
```

```
      \        )\n          +\n          sum(rate(apiserver_request_total{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",code=~\"5..\"}[5m]))\n        )\n        /\n\
      \        sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\"}[5m]))\n      labels:\n        verb: write\n      record:\
      \ apiserver_request:burnrate5m\n    - expr: |2\n        (\n          (\n   \
      \         # too slow\n
sum(rate(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\"}[6h]))\n           \
      \ -\n            sum(rate(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\",le=\"1\"}[6h]))\n          )\n          +\n\
      \          sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"\
      POST|PUT|PATCH|DELETE\",code=~\"5..\"}[6h]))\n        )\n        /\n       \
      \ sum(rate(apiserver_request_total{job=\"kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|
DELETE\"\
      }[6h]))\n      labels:\n        verb: write\n      record:
apiserver_request:burnrate6h\n\
      \    - expr: |\n        sum by (code,resource) (rate(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[5m]))\n      labels:\n        verb:\
      \ read\n      record: code_resource:apiserver_request_total:rate5m\n    - expr:\
      \ |\n        sum by (code,resource) (rate(apiserver_request_total{job=\"kubernetes-
apiservers\"\
      ,verb=~\"POST|PUT|PATCH|DELETE\"}[5m]))\n      labels:\n        verb: write\n\
      \      record: code_resource:apiserver_request_total:rate5m\n    - expr: |\n\
      \        histogram_quantile(0.99, sum by (le, resource)
(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[5m]))) > 0\n      labels:\n    \
      \    quantile: \"0.99\"\n        verb: read\n      record:
cluster_quantile:apiserver_request_duration_seconds:histogram_quantile\n\
      \    - expr: |\n        histogram_quantile(0.99, sum by (le, resource)
(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"POST|PUT|PATCH|DELETE\"}[5m]))) > 0\n      labels:\n\
      \        quantile: \"0.99\"\n        verb: write\n      record:
cluster_quantile:apiserver_request_duration_seconds:histogram_quantile\n\
      \    - expr: |2\n        sum(rate(apiserver_request_duration_seconds_sum{subresource!
=\"\
      log\",verb!~\"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT\"}[5m]))
without(instance,\
      \ pod)\n        /\n
sum(rate(apiserver_request_duration_seconds_count{subresource!=\"\
      log\",verb!~\"LIST|WATCH|WATCHLIST|DELETECOLLECTION|PROXY|CONNECT\"}[5m]))
without(instance,\
      \ pod)\n      record: cluster:apiserver_request_duration_seconds:mean5m\n  \
      \  - expr: |\n        histogram_quantile(0.99,
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",subresource!=\"log\",verb!~\"LIST|WATCH|WATCHLIST|
DELETECOLLECTION|PROXY|CONNECT\"\
      }[5m])) without(instance, pod))\n      labels:\n        quantile: \"0.99\"\n\
      \      record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile\n\
      \    - expr: |\n        histogram_quantile(0.9,
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",subresource!=\"log\",verb!~\"LIST|WATCH|WATCHLIST|
DELETECOLLECTION|PROXY|CONNECT\"\
      }[5m])) without(instance, pod))\n      labels:\n        quantile: \"0.9\"\n\
```

```
      \      record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile\n\
      \    - expr: |\n        histogram_quantile(0.5,
sum(rate(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",subresource!=\"log\",verb!~\"LIST|WATCH|WATCHLIST|
DELETECOLLECTION|PROXY|CONNECT\"\
      }[5m])) without(instance, pod))\n      labels:\n        quantile: \"0.5\"\n\
      \      record: cluster_quantile:apiserver_request_duration_seconds:histogram_quantile\n\
      \  - interval: 3m\n    name: kube-apiserver-availability.rules\n    rules:\n\
      \    - expr: |2\n        1 - (\n          (\n            # write too slow\n
      \            sum(increase(apiserver_request_duration_seconds_count{verb=~\"\
      POST|PUT|PATCH|DELETE\"}[30d]))\n            -\n
sum(increase(apiserver_request_duration_seconds_bucket{verb=~\"\
      POST|PUT|PATCH|DELETE\",le=\"1\"}[30d]))\n          ) +\n          (\n     \
      \       # read too slow\n
sum(increase(apiserver_request_duration_seconds_count{verb=~\"\
      LIST|GET\"}[30d]))\n            -\n            (\n              (\n        \
      \        sum(increase(apiserver_request_duration_seconds_bucket{verb=~\"LIST|GET\"\
      ,scope=~\"resource|\",le=\"0.1\"}[30d]))\n                or\n             \
      \   vector(0)\n              )\n              +\n
sum(increase(apiserver_request_duration_seconds_bucket{verb=~\"\
      LIST|GET\",scope=\"namespace\",le=\"0.5\"}[30d]))\n              +\n       \
      \       sum(increase(apiserver_request_duration_seconds_bucket{verb=~\"LIST|GET\"\
      ,scope=\"cluster\",le=\"5\"}[30d]))\n            )\n          ) +\n        \
      \  # errors\n          sum(code:apiserver_request_total:increase30d{code=~\"\
      5..\"} or vector(0))\n        )\n        /\n
sum(code:apiserver_request_total:increase30d)\n\
      \      labels:\n        verb: all\n      record: apiserver_request:availability30d\n\
      \    - expr: |2\n        1 - (\n
sum(increase(apiserver_request_duration_seconds_count{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\"}[30d]))\n          -\n          (\n\
      \            # too slow\n            (\n
sum(increase(apiserver_request_duration_seconds_bucket{job=\"\
      kubernetes-apiservers\",verb=~\"LIST|GET\",scope=~\"resource|\",le=\"0.1\"}[30d]))\n\
      \              or\n              vector(0)\n            )\n            +\n \
      \            sum(increase(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"LIST|GET\",scope=\"namespace\",le=\"0.5\"}[30d]))\n            +\n\
      \            sum(increase(apiserver_request_duration_seconds_bucket{job=\"kubernetes-
apiservers\"\
      ,verb=~\"LIST|GET\",scope=\"cluster\",le=\"5\"}[30d]))\n          )\n      \
      \    +\n          # errors\n
sum(code:apiserver_request_total:increase30d{verb=\"\
      read\",code=~\"5..\"} or vector(0))\n        )\n        /\n
sum(code:apiserver_request_total:increase30d{verb=\"\
      read\"})\n      labels:\n        verb: read\n      record:
apiserver_request:availability30d\n\
      \    - expr: |2\n        1 - (\n          (\n            # too slow\n      \
      \      sum(increase(apiserver_request_duration_seconds_count{verb=~\"POST|PUT|PATCH|
DELETE\"\
      }[30d]))\n            -\n
sum(increase(apiserver_request_duration_seconds_bucket{verb=~\"\
      POST|PUT|PATCH|DELETE\",le=\"1\"}[30d]))\n          )\n          +\n       \
      \   # errors\n          sum(code:apiserver_request_total:increase30d{verb=\"\
      write\",code=~\"5..\"} or vector(0))\n        )\n        /\n
sum(code:apiserver_request_total:increase30d{verb=\"\
```

```
      write\"})\n      labels:\n        verb: write\n      record:
apiserver_request:availability30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"LIST\",code=~\"2..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"GET\",code=~\"2..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"POST\",code=~\"2..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PUT\",code=~\"2..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PATCH\",code=~\"2..\"}[30d]))\n      record:\
      \ code_verb:apiserver_request_total:increase30d\n    - expr: |\n        sum\
      \ by (code, verb) (increase(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=\"DELETE\",code=~\"2..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"LIST\",code=~\"3..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"GET\",code=~\"3..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"POST\",code=~\"3..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PUT\",code=~\"3..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PATCH\",code=~\"3..\"}[30d]))\n      record:\
      \ code_verb:apiserver_request_total:increase30d\n    - expr: |\n        sum\
      \ by (code, verb) (increase(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=\"DELETE\",code=~\"3..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"LIST\",code=~\"4..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"GET\",code=~\"4..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"POST\",code=~\"4..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PUT\",code=~\"4..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PATCH\",code=~\"4..\"}[30d]))\n      record:\
      \ code_verb:apiserver_request_total:increase30d\n    - expr: |\n        sum\
      \ by (code, verb) (increase(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=\"DELETE\",code=~\"4..\"}[30d]))\n      record:
```

```
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"LIST\",code=~\"5..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"GET\",code=~\"5..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"POST\",code=~\"5..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PUT\",code=~\"5..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code, verb) (increase(apiserver_request_total{job=\"\
      kubernetes-apiservers\",verb=\"PATCH\",code=~\"5..\"}[30d]))\n      record:\
      \ code_verb:apiserver_request_total:increase30d\n    - expr: |\n        sum\
      \ by (code, verb) (increase(apiserver_request_total{job=\"kubernetes-apiservers\"\
      ,verb=\"DELETE\",code=~\"5..\"}[30d]))\n      record:
code_verb:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code)
(code_verb:apiserver_request_total:increase30d{verb=~\"\
      LIST|GET\"})\n      labels:\n        verb: read\n      record:
code:apiserver_request_total:increase30d\n\
      \    - expr: |\n        sum by (code)
(code_verb:apiserver_request_total:increase30d{verb=~\"\
      POST|PUT|PATCH|DELETE\"})\n      labels:\n        verb: write\n      record:\
      \ code:apiserver_request_total:increase30d\n"
    rules_yml: '{}

      '
  deployment:
    configmapReload:
      containers:
        args:
        - --volume-dir=/etc/config
        - --webhook-url=http://127.0.0.1:9090/-/reload
        resources: {}
    containers:
      args:
      - --storage.tsdb.retention.time=42d
      - --config.file=/etc/config/prometheus.yml
      - --storage.tsdb.path=/data
      - --web.console.libraries=/etc/prometheus/console_libraries2
      - --web.console.templates=/etc/prometheus/consoles
      - --web.enable-lifecycle
      resources: {}
    podAnnotations: {}
    podLabels: {}
    replicas: 1
    rollingUpdate:
      maxSurge: null
      maxUnavailable: null
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
```

```
    annotations: {}
    storage: 150Gi
    storageClassName: wcpglobalstorageprofile
  service:
    annotations: {}
    labels: {}
    port: 80
    targetPort: 9090
    type: ClusterIP
pushgateway:
  deployment:
    containers:
      resources: {}
    podAnnotations: {}
    podLabels: {}
    replicas: 1
  service:
    annotations: {}
    labels: {}
    port: 9091
    targetPort: 9091
    type: ClusterIP
```

## prometheus.yaml

La especificación `prometheus.yaml` hace referencia al secreto de `prometheus-data-values`.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus-sa
  namespace: tkg-system

---
# temp
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: prometheus-sa
    namespace: tkg-system

---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: prometheus
  namespace: tkg-system
spec:
```

```
serviceAccountName: prometheus-sa
packageRef:
  refName: prometheus.tanzu.vmware.com
  versionSelection:
    constraints: 2.45.0+vmware.1-tkg.2
values:
- secretRef:
    name: prometheus-data-values
```

# Instalar Grafana en TKr para vSphere 7.x

Consulte estas instrucciones para instalar Grafana en un clúster de TKG aprovisionado con TKr para vSphere 7.x.

## Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

## Instalar Grafana

Instale Grafana.

1 Enumere las versiones de Grafana disponibles en el repositorio.

```
kubectl get packages -n tkg-system | grep grafana
```

2 Cree el espacio de nombres de Grafana.

```
kubectl create ns tanzu-system-dashboards
```

3 Cree una etiqueta de PSA para el espacio de nombres.

```
kubectl label namespace tanzu-system-dashboards pod-security.kubernetes.io/
enforce=privileged
```

4 Como alternativa, cree de forma declarativa la etiqueta y el espacio de nombres de Grafana mediante el archivo `ns-grafana-dashboard.yaml`.

```
apiVersion:
v1kind: Namespace
metadata:
  name: grafana-dashboard
---
apiVersion: v1
kind: Namespace
metadata:
  name: tanzu-system-dashboards
  labels:
    pod-security.kubernetes.io/enforce: privileged
```

5 Cree `grafana-data-values.yaml`.

Consulte Referencia del paquete de Grafana.

6 Cree un secreto con el archivo `grafana-data-values.yaml` como entrada.

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-
values.yaml -n tkg-system
```

```
secret/grafana-data-values created
```

7 Compruebe el secreto.

```
kubectl get secrets -A
```

```
kubectl describe secret grafana-data-values -n tkg-system
```

8 Si es necesario, personalice `grafana-data-values` en función de su entorno.

Consulte Referencia del paquete de Grafana.

Si actualiza los valores de datos, actualice el secreto con el siguiente comando.

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-
values.yaml -n tkg-system -o yaml --dry-run=client | kubectl replace -f-
```

```
secret/grafana-data-values replaced
```

9 Cree la especificación `grafana.yaml`.

Consulte Instalar Grafana en TKr para vSphere 7.x.

10 Instale Grafana.

```
kubectl apply -f grafana.yaml
```

```
serviceaccount/grafana-sa created
clusterrolebinding.rbac.authorization.k8s.io/grafana-role-binding created
packageinstall.packaging.carvel.dev/grafana created
```

11 Compruebe la instalación del paquete de Grafana.

```
kubectl get pkgi -A | grep grafana
```

12 Compruebe los objetos de Grafana.

```
kubectl get all -n tanzu-system-dashboards
```

## Acceder al panel de control de Grafana mediante Envoy LoadBalancer

Si se implementa el servicio Contour Envoy de tipo LoadBalancer como requisito previo y lo especificó en el archivo de configuración de Grafana, obtenga la dirección IP externa del equilibrador de carga y cree registros de DNS para el FQDN de Grafana.

1 Obtenga la dirección `External-IP` para el servicio Envoy de tipo LoadBalancer.

```
kubectl get service envoy -n tanzu-system-ingress
```

Debería ver la dirección `External-IP` que se devuelve, por ejemplo:

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

Si lo prefiere, puede obtener la dirección `External-IP` mediante el siguiente comando.

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Para comprobar la instalación de la extensión Grafana, actualice el archivo `/etc/hosts` local con el FQDN de Grafana asignado a la dirección `External-IP` del equilibrador de carga, como por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
#TKG Grafana Extension with Envoy Load Balancer
10.195.141.17 grafana.system.tanzu
```

3 Para acceder al panel de control de Grafana, desplácese hasta `https://grafana.system.tanzu`.

Dado que el sitio utiliza certificados autofirmados, es posible que tenga que pasar por una advertencia de seguridad específica del navegador antes de poder acceder al panel de control.

4 Para el acceso de producción, cree dos registros CNAME en un servidor DNS que asignen la dirección `External-IP` del equilibrador de carga del servicio Envoy al panel de control de Grafana.

## Acceder al panel de control de Grafana mediante NodePort de Envoy

Si se implementa el servicio Contour Envoy de tipo NodePort como requisito previo y lo especificó en el archivo de configuración de Grafana, obtenga la dirección IP de la máquina virtual de un nodo de trabajo y cree registros de DNS para el FQDN de Grafana.

1 Cambie el contexto a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster.

```
kubectl config use-context VSPHERE-NAMESPACE
```

2 Enumere los nodos del clúster.

```
kubectl get virtualmachines
```

3 Seleccione uno de los nodos de trabajo y describa el nodo mediante el siguiente comando.

```
kubectl describe virtualmachines tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 Busque la dirección IP de la máquina virtual; por ejemplo, `Vm Ip: 10.115.22.43`.

5 Para comprobar la instalación de la extensión Grafana, actualice el archivo `/etc/hosts` local con el FQDN de Grafana asignado a una dirección IP de nodo de trabajo, como por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKG Grafana with Envoy NodePort
10.115.22.43 grafana.system.tanzu
```

6 Para acceder al panel de control de Grafana, desplácese hasta `https://grafana.system.tanzu`.

Dado que el sitio utiliza certificados autofirmados, es posible que tenga que pasar por una advertencia de seguridad específica del navegador antes de poder acceder al panel de control.

## grafana-data-values.yaml

Consulte el archivo `grafana-data-values.yaml` de ejemplo.

```
namespace: tanzu-system-dashboards
grafana:
  pspNames: "vmware-system-restricted"
  deployment:
    replicas: 1
    updateStrategy: Recreate
  pvc:
    accessMode: ReadWriteOnce
    storage: 2Gi
    storageClassName: wcpglobalstorageprofile
  secret:
    admin_user: YWRtaW4=
    admin_password: YWRtaW4=
    type: Opaque
  service:
    port: 80
    targetPort: 3000
    type: LoadBalancer
ingress:
  enabled: true
  prefix: /
  servicePort: 80
  virtual_host_fqdn: grafana.system.tanzu
```

### grafana.yaml

Consulte la especificación `grafana.yaml` de ejemplo. Actualice la versión del paquete según sea necesario.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: grafana-sa
  namespace: tkg-system
  annotations:
    pod-security.kubernetes.io/enforce: "privileged"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: grafana-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: grafana-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: grafana
  namespace: tkg-system
spec:
  serviceAccountName: grafana-sa
  packageRef:
    refName: grafana.tanzu.vmware.com
    versionSelection:
      constraints: 10.0.1+vmware.1-tkg.2 #PKG-VERSION
  values:
  - secretRef:
      name: grafana-data-values
```

## Instalar Harbor en TKr para vSphere 7.x

Siga estas instrucciones para instalar Harbor en un clúster de TKG aprovisionado con un TKr para vSphere 7.x.

### Requisitos previos

Consulte Flujo de trabajo para instalar paquetes estándar en TKr para vSphere 7.x.

Harbor requiere una entrada HTTP/S. Los servicios Harbor se exponen a través de un servicio Envoy en el paquete Contour. Como requisito previo, implemente el paquete de Contour. Consulte Instalar Contour en TKr para vSphere 7.x.

- Si utiliza redes de NSX para Supervisor, cree un servicio Envoy de tipo LoadBalancer.
- Si utiliza redes de vSphere vDS para Supervisor, cree un servicio Envoy de tipo LoadBalancer o NodePort, en función de cuáles sean el entorno y los requisitos.

La extensión Harbor requiere de DNS. Para fines de realización de pruebas y verificación, agregue los FQDN de Harbor y Notary al archivo local `/etc/hosts`. Las instrucciones que aparecen a continuación describen cómo hacerlo.

En la fase de producción, Harbor requiere una zona DNS en un servidor DNS local, como BIND, o en una nube pública, como AWS Route53 o Azure DNS. Una vez que haya configurado DNS, instale la extensión ExternalDNS si desea registrar automáticamente los FQDN de Harbor con un servidor DNS. Consulte Instalar ExternalDNS en TKr para vSphere 7.x.

## Instalar Harbor

Para instalar el registro de Harbor mediante el paquete estándar, complete los siguientes pasos.

1 Enumere las versiones de Harbor disponibles en el repositorio.

```
kubectl get packages -n tkg-system | grep harbor
```

2 Cree la especificación `harbor.yaml`.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: harbor-sa
  namespace: tkg-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: habor-role-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
  - kind: ServiceAccount
    name: harbor-sa
    namespace: tkg-system
---
apiVersion: packaging.carvel.dev/v1alpha1
kind: PackageInstall
metadata:
  name: harbor
  namespace: tkg-system
spec:
  serviceAccountName: harbor-sa
```

```
  packageRef:
    refName: harbor.tanzu.vmware.com
    versionSelection:
      constraints: 2.7.1+vmware.1-tkg.1 #PKG-VERSION
  values:
  - secretRef:
      name: harbor-data-values
---
apiVersion: v1
kind: Secret
metadata:
  name: harbor-data-values
  namespace: harbor-registry
stringData:
  values.yml: |
    namespace: tanzu-system-registry
    hostname: <ENTER-HARBOR-FQDN>
    port:
      https: 443
    logLevel: info
    tlsCertificate:
      tls.crt: ""
      tls.key: ""
      ca.crt:
    tlsCertificateSecretName:
    enableContourHttpProxy: true
    harborAdminPassword: <ENTER-STRONG-PASSWORD-HERE>
    secretKey: <ENTER-SECRET-KEY>
    database:
      password: <ENTER-STRONG-PASSWORD-HERE>
      shmSizeLimit:
      maxIdleConns:
      maxOpenConns:
    exporter:
      cacheDuration:
    core:
      replicas: 1
      secret: <ENTER-SECRET>
      xsrfKey: <ENTER-XSRF-KEY-WHICH-IS-AN-ALPHANUMERIC-STRING-WITH-32-CHARS>
    jobservice:
      replicas: 1
      secret: <ENTER-SECRET>
    registry:
      replicas: 1
      secret: <ENTER-SECRET>
    trivy:
      enabled: true
      replicas: 1
      gitHubToken: ""
      skipUpdate: false
    persistence:
      persistentVolumeClaim:
        registry:
          existingClaim: ""
          storageClass: "<ENTER-STORAGE-CLASS>"
```

```
          subPath: ""
          accessMode: ReadWriteOnce
          size: 50Gi
        jobservice:
          existingClaim: ""
          storageClass: "<ENTER-STORAGE-CLASS>"
          subPath: ""
          accessMode: ReadWriteOnce
          size: 10Gi
        database:
          existingClaim: ""
          storageClass: "<ENTER-STORAGE-CLASS>"
          subPath: ""
          accessMode: ReadWriteOnce
          size: 10Gi
        redis:
          existingClaim: ""
          storageClass: "<ENTER-STORAGE-CLASS>"
          subPath: ""
          accessMode: ReadWriteOnce
          size: 10Gi
        trivy:
          existingClaim: ""
          storageClass: "<ENTER-STORAGE-CLASS>"
          subPath: ""
          accessMode: ReadWriteOnce
          size: 10Gi
    proxy:
      httpProxy:
      httpsProxy:
      noProxy: 127.0.0.1,localhost,.local,.internal
    pspNames: vmware-system-restricted
    network:
      ipFamilies: ["IPv4", "IPv6"]
```

3 Personalice el secreto `harbor-data-values` en la especificación `harbor.yaml` con los valores adecuados para su entorno, incluidos el nombre de host, las contraseñas, los secretos y la clase de almacenamiento.

   Para obtener más instrucciones, consulte Referencia del paquete de Harbor.

4 Instale Harbor.

```
kubectl apply -f harbor.yaml
```

5 Compruebe la instalación de Harbor.

```
kubectl get all -n harbor-registry
```

## Configurar DNS para Harbor mediante Envoy LoadBalancer (redes NSX)

Si el servicio Envoy de requisitos previos se expone a través de LoadBalancer, obtenga la dirección IP externa del equilibrador de carga y cree registros de DNS para los FQDN de Harbor.

1 Obtenga la dirección `External-IP` para el servicio Envoy de tipo LoadBalancer.

```
kubectl get service envoy -n tanzu-system-ingress
```

Debería ver la dirección `External-IP` que se devuelve, por ejemplo:

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

Si lo prefiere, puede obtener la dirección `External-IP` mediante el siguiente comando.

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Para comprobar la instalación de la extensión Harbor, actualice el archivo `/etc/hosts` local con los FQDN de Harbor y Notary asignados a la dirección `External-IP` del equilibrador de carga; por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
#TKG Harbor with Envoy Load Balancer IP
10.195.141.17 core.harbor.domain
10.195.141.17 core.notary.harbor.domain
```

3 Para comprobar la instalación de la extensión Harbor, inicie sesión en Harbor.

4 Cree dos registros CNAME en un servidor DNS que asignen la dirección `External-IP` del servicio Envoy del equilibrador de carga al FQDN de Harbor y al FQDN de Notary.

5 Instale la extensión DNS externo.

## Configurar DNS para Harbor mediante Envoy NodePort (redes vDS)

Si el servicio Envoy de requisitos previos se expone a través de NodePort, obtenga la dirección IP de la máquina virtual de un nodo de trabajo y cree registros de DNS para los FQDN de Harbor.

**Nota** Para usar NodePort, debe haber especificado el valor de `port.https` correcto en el archivo `harbor-data-values.yaml`.

1 Cambie el contexto a la instancia de espacio de nombres de vSphere en la que se aprovisiona el clúster.

```
kubectl config use-context VSPHERE-NAMESPACE
```

2 Enumere los nodos del clúster.

```
kubectl get virtualmachines
```

3 Seleccione uno de los nodos de trabajo y describa el nodo mediante el siguiente comando.

```
kubectl describe virtualmachines tkg2-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 Busque la dirección IP de la máquina virtual; por ejemplo, `Vm Ip: 10.115.22.43`.

5 Para comprobar la instalación de la extensión Harbor, actualice el archivo `/etc/hosts` local con los FQDN de Harbor y Notary asignados a la dirección IP del nodo de trabajo; por ejemplo:

```
127.0.0.1 localhost
127.0.1.1 ubuntu
#TKG Harbor with Envoy NodePort
10.115.22.43 core.harbor.domain
10.115.22.43 core.notary.harbor.domain
```

6 Para comprobar la instalación de la extensión Harbor, inicie sesión en Harbor.

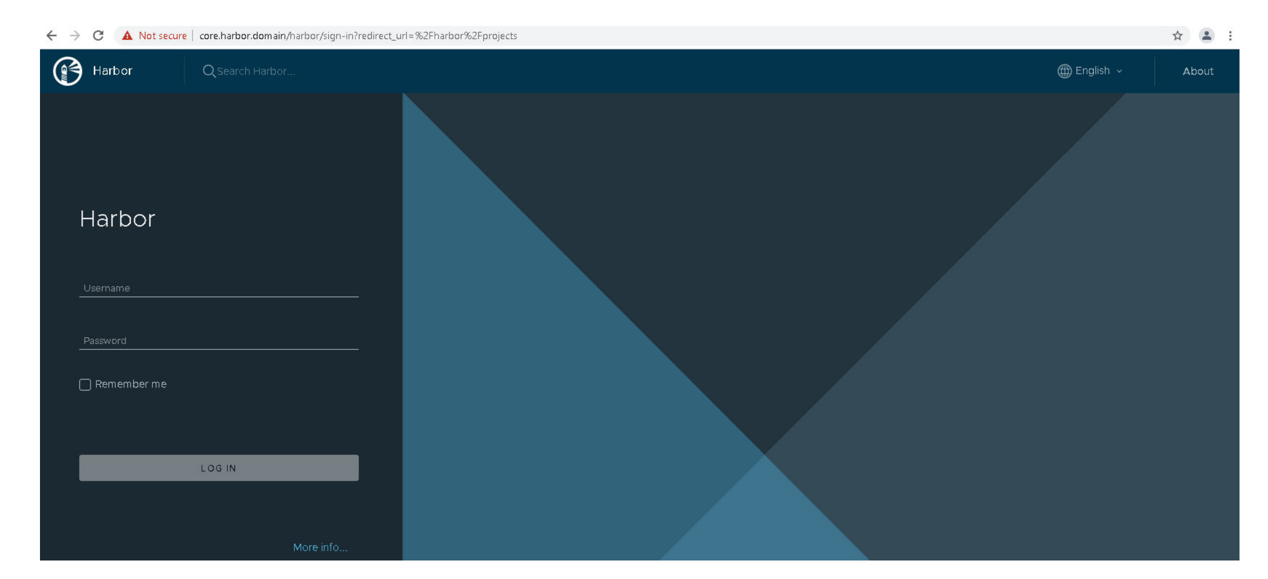
7 Cree dos registros CNAME en un servidor DNS que asignen la dirección IP del nodo de trabajo al FQDN de Harbor y al FQDN de Notary.

8 Instale la extensión DNS externo.

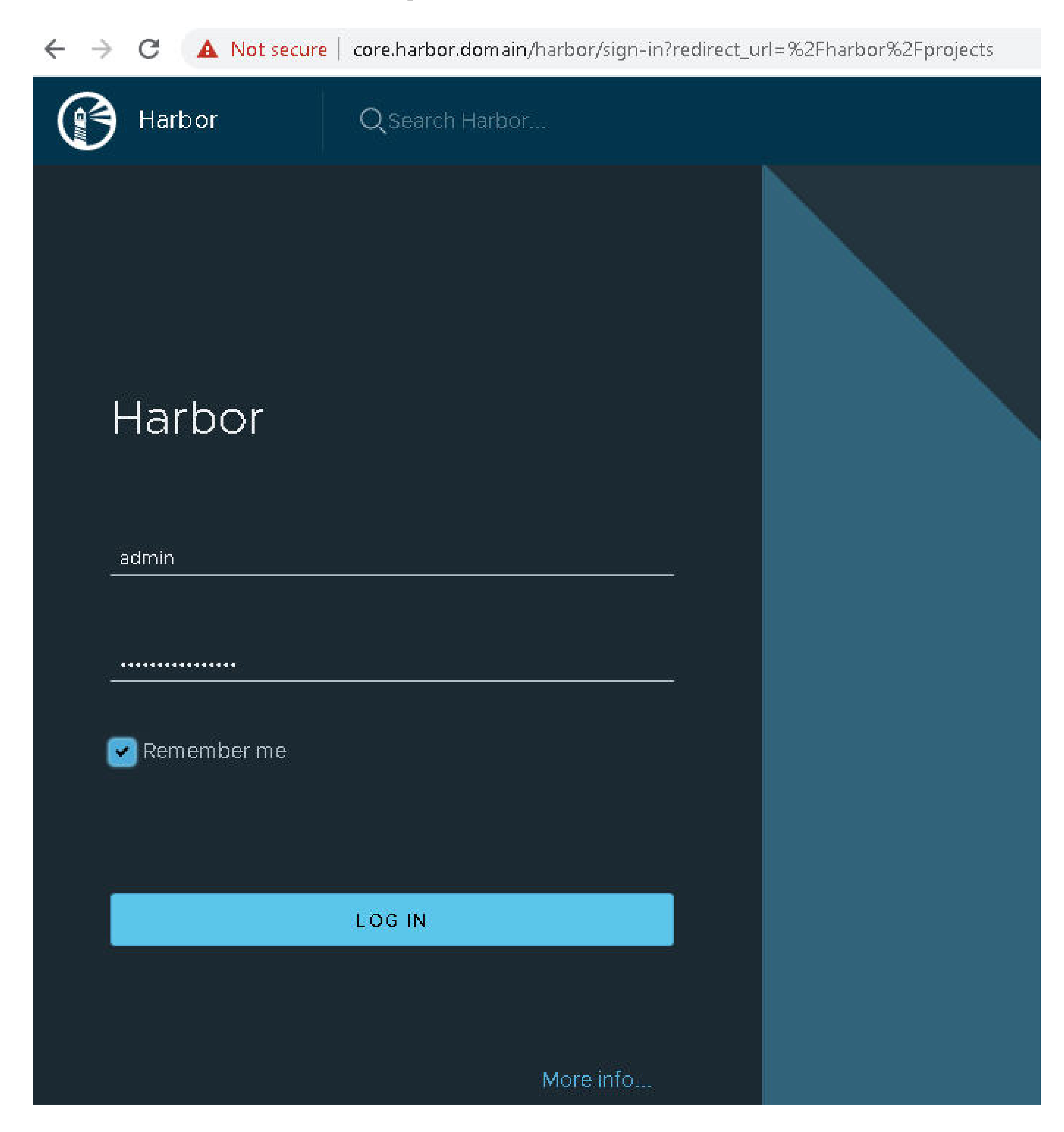
## Iniciar sesión en la interfaz web de Harbor

Una vez que Harbor esté instalado y configurado, inicie sesión y comience a utilizarlo.

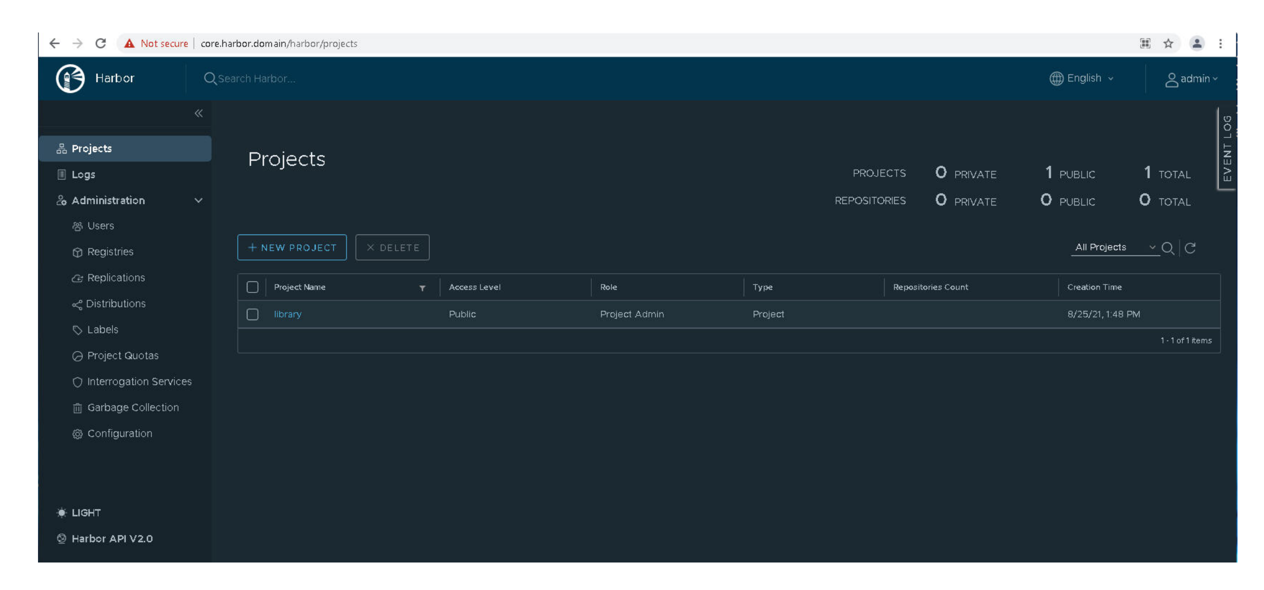
1 Acceda a la interfaz web del registro de Harbor en `https://core.harbor.domain` o al nombre de host que utilizó.



2 Inicie sesión en Harbor con el nombre de usuario **`admin`** y la contraseña generada que colocó en el archivo `harbor-data-values.yaml`.




3 Compruebe que puede acceder a la interfaz de usuario de Harbor.




4 Obtenga el certificado de CA de Harbor.

En la interfaz de Harbor, seleccione **Proyectos > biblioteca** o cree un **Nuevo proyecto**.

Haga clic en **Certificado del registro** y descargue el certificado de CA de Harbor (`ca.crt`).

5 Agregue el certificado de CA de Harbor al almacén de confianza del cliente de Docker para poder insertar y extraer imágenes de contenedor al registro de Harbor y desde él. Consulte Capítulo 14 Uso de registros privados con clústeres Servicio TKG.

6 Consulte la documentación de Harbor para obtener más información sobre el uso de Harbor.

# Implementar cargas de trabajo en clústeres de Servicio TKG

12

Puede implementar cargas de trabajo de aplicaciones en clústeres de Servicio TKG mediante pods, servicios, volúmenes persistentes y recursos de nivel superior, como implementaciones y conjuntos de réplicas.

Lea los siguientes temas a continuación:

- Implementación de pods con el servicio de equilibrador de carga
- Servicio de equilibrador de carga con IP estática
- Entrada mediante Nginx
- Entrada mediante Contour
- Usar clases de almacenamiento para volúmenes persistentes
- Crear volúmenes de almacenamiento persistentes de forma dinámica
- Crear volúmenes de almacenamiento persistentes de forma estática
- Implementar la aplicación del libro de visitas en un clúster de TKG
- YAML de la aplicación del libro de visitas
- Implementar la aplicación StatefulSet en Zonas de vSphere con asociación de volúmenes de enlace en tiempo de ejecución

## Implementación de pods con el servicio de equilibrador de carga

Para enrutar el tráfico externo a los pods que se ejecutan en un clúster de TKG 2.0, cree un servicio de tipo LoadBalancer. El servicio de equilibrador de carga expone una dirección IP pública desde la cual se enruta el tráfico entrante a los pods.

Puede aprovisionar un equilibrador de carga externo para los pods de Kubernetes que se exponen como servicios. Por ejemplo, puede implementar un contenedor de Nginx y exponerlo como servicio de Kubernetes de tipo LoadBalancer.

#### Requisitos previos

- Revise el tipo de servicio LoadBalancer en la documentación de Kubernetes.
- Aprovisione un clúster de TKG.

- Conéctese al clúster de TKG.

### Procedimiento

1 Cree el siguiente archivo YAML `nginx-lbsvc.yaml`.

Este archivo YAML define un servicio de Kubernetes de tipo LoadBalancer e implementa un contenedor de Nginx como un equilibrador de carga externo para el servicio.

```
kind: Service
apiVersion: v1
metadata:
  name: srvclb-ngnx
spec:
  selector:
    app: hello
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: loadbalancer
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: "nginxdemos/hello"
```

2 Aplique el archivo YAML.

```
kubectl apply -f nginx-lbsvc.yaml
```

3 Compruebe la implementación del servicio nginx.

```
kubectl get services
```

`srvclb-ngnx` tiene una dirección IP externa e interna.

```
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
srvclb-ngnx   LoadBalancer   10.11.12.19      10.19.15.89     80:30818/TCP   18m
```

4 Desde un navegador, introduzca la dirección IP externa del servicio LoadBalancer de Nginx.

Verá un mensaje con el banner de `NGINX` y los detalles del equilibrador de carga.

# Servicio de equilibrador de carga con IP estática

Puede configurar un servicio de Kubernetes de tipo LoadBalancer para que utilice una dirección IP estática. Tenga en cuenta los requisitos de componente mínimos, un importante aspecto sobre seguridad y las instrucciones de fortalecimiento del clúster cuando implemente esta función.

## Usar una IP estática en un servicio de tipo LoadBalancer

Por lo general, cuando se define un servicio de Kubernetes de Tipo LoadBalancer, el equilibrador de carga asigna una dirección IP efímera.

También puede especificar una dirección IP estática para el equilibrador de carga. Al crear el servicio, la instancia del equilibrador de carga se aprovisiona con la dirección IP estática que asignó.

El siguiente servicio de ejemplo demuestra cómo configurar un equilibrador de carga compatible con una dirección IP estática. En la especificación de servicio, se incluyen el parámetro `loadBalancerIP` y un valor de dirección IP, el cual es `10.11.12.49` en este ejemplo.

```
kind: Service
apiVersion: v1
metadata:
  name: load-balancer-service-with-static-ip
spec:
  selector:
    app: hello-world
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
  loadBalancerIP: 10.11.12.49
```

Para NSX Advanced Load Balancer, utilice una dirección IP del grupo de IPAM que se configuró para el equilibrador de carga cuando se instaló. Cuando se crea el servicio y se asigna la dirección IP estática, el equilibrador de carga la marca como asignada y administra el ciclo de vida de la dirección IP de la misma forma que una dirección IP efímera. Es decir, si se elimina el servicio, la dirección IP no está asignada y se vuelve disponible para reasignarla.

En el caso del equilibrador de carga de NSX-T, tiene dos opciones. El mecanismo predeterminado es el mismo que en NSX Advanced Load Balancer: utilice una dirección IP tomada del grupo de direcciones IP que se configuró para el equilibrador de carga cuando se instaló. Cuando se asigna la dirección IP estática, el equilibrador de carga la marca automáticamente como asignada y administra su ciclo de vida.

La segunda opción de NSX-T consiste en preasignar manualmente la dirección IP estática. En este caso, se utiliza una dirección IP que no forma parte del grupo de direcciones IP del equilibrador de carga externo asignado al equilibrador de carga, sino que se toma de un grupo de direcciones IP flotantes. En este caso, administra manualmente la asignación y el ciclo de vida de la dirección IP mediante NSX Manager.

## Requisitos importantes de fortalecimiento y consideración de seguridad

Puede producirse un problema de seguridad que se debe tener en cuenta al utilizar esta función. Si un desarrollador puede revisar el valor de `Service.status.loadBalancerIP`, es posible que el desarrollador pueda secuestrar el tráfico en el clúster destinado para la dirección IP con revisiones. En concreto, si una función o ClusterRole con el permiso `patch` está enlazada a una cuenta de usuario o servicio en un clúster donde esté implementada esta función, ese propietario de cuenta puede usar sus propias credenciales para emitir comandos `kubectl` y cambiar la dirección IP estática asignada al equilibrador de carga.

Para evitar las posibles implicaciones de seguridad que tiene usar la asignación de direcciones IP estáticas para un servicio de equilibrador de carga, debe fortalecer cada clúster en el que vaya a implementar esta función. Para ello, la función o ClusterRole que defina para cualquier desarrollador no debe permitir el verbo `patch` para `apiGroups:""` y `resources: services/status`. El fragmento de función de ejemplo demuestra lo que no se debe hacer al implementar esta función.

NO PERMITIR LA REVISIÓN

```
- apiGroups:
  - ""
  resources:
  - services/status
  verbs:
  - patch
```

Para comprobar si un desarrollador tiene permisos de revisión, ejecute el siguiente comando como ese usuario:

```
kubectl --kubeconfig <KUBECONFIG> auth can-i patch service/status
```

Si el comando devuelve `yes`, el usuario tiene permisos de revisión. Consulte Comprobación de acceso a la API (Checking API Access) en la documentación de Kubernetes para obtener más información.

Para conceder al desarrollador acceso a un clúster, consulte .Conceder a los desarrolladores acceso de vCenter SSO a los clústeres de Servicio TKG. Para obtener una plantilla de función de ejemplo que pueda personalizar, consulte Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG. Para obtener un ejemplo sobre cómo restringir el acceso al clúster, consulte https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-example.

# Entrada mediante Nginx

Un recurso de entrada de Kubernetes proporciona enrutamiento HTTP o HTTPS desde fuera del clúster a uno o varios servicios dentro del clúster. Los clústeres de TKG admiten la entrada a través de controladoras de terceros, como Nginx.

En este tutorial, se demuestra cómo implementar un servicio de entrada de Kubernetes basado en Nginx para enrutar el tráfico externo a los servicios en un clúster de Tanzu Kubernetes. Un servicio de entrada requiere una controladora de entrada. Instalamos la controladora de entrada de Nginx con Helm. Helm es un administrador de paquetes para Kubernetes.

---

**Nota** Existen varias formas de llevar a cabo esta tarea. Los pasos que se indican a continuación corresponden a un enfoque. Otros enfoques pueden ser más adecuados para su entorno en particular.

---

#### Requisitos previos

- Revise el recurso de Entrada en la documentación de Kubernetes.
- Revise la documentación de la controladora de entrada de Nginx.
- Aprovisione un clúster de TKG.
- Habilite la directiva de seguridad de pods, si es necesario.
- Conéctese al clúster de TKG.

#### Procedimiento

1 Para instalar Helm, consulte la documentación.

2 Instale la controladora de entrada de Nginx con Helm.

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm install ingress-nginx ingress-nginx/ingress-nginx
```

3 Compruebe que la controladora de entrada de Nginx esté implementada como un servicio de tipo equilibrador de carga (LoadBalancer).

```
kubectl get services
```

```
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.16.18.20    10.19.14.76    80:30635/
TCP,443:30873/TCP   59m
ingress-nginx-controller-admission   ClusterIP      10.87.41.25    <none>
443/TCP                      59m
```

4 Ejecute ping en el equilibrador de carga utilizando la dirección IP externa.

```
ping 10.19.14.76
```

```
Pinging 10.19.14.76 with 32 bytes of data:
Reply from 10.19.14.76: bytes=32 time<1ms TTL=62
Reply from 10.19.14.76: bytes=32 time=1ms TTL=62
```

5 Compruebe que la controladora de entrada de Nginx esté en ejecución.

```
kubectl get pods
```

```
NAME                                        READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-7c6c46898c-v6blt   1/1     Running   0          76m
```

6 Cree un recurso de entrada con una regla de entrada y una ruta de acceso denominada `ingress-hello.yaml`.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-hello
spec:
  rules:
  - http:
      paths:
      - path: /hello
        backend:
          serviceName: hello
          servicePort: 80
```

**Atención** La API de Kubernetes `networking.k8s.io/v1beta1` está obsoleta desde Kubernetes v1.22 en adelante. La nueva API que se utilizará es `networking.k8s.io/v1` e implica varios cambios en el manifiesto que se documentan aquí https://kubernetes.io/docs/reference/using-api/deprecation-guide/.

7 Implemente el recurso `ingress-hello`.

```
kubectl apply -f ingress-hello.yaml
```

```
ingress.networking.k8s.io/ingress-hello created
```

8 Compruebe que el recurso de entrada se haya implementado.

Tenga en cuenta que la dirección IP se asigna a la IP externa de la controladora de entrada.

```
kubectl get ingress
```

```

NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      51m
```

9 Cree una aplicación y un servicio de prueba Hello denominados `ingress-hello-test.yaml`.

```

kind: Service
apiVersion: v1
metadata:
  name: hello
spec:
  selector:
    app: hello
    tier: backend
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello
      tier: backend
      track: stable
  template:
    metadata:
      labels:
        app: hello
        tier: backend
        track: stable
    spec:
      containers:
        - name: hello
```

```
          image: "gcr.io/google-samples/hello-go-gke:1.0"
          ports:
            - name: http
              containerPort: 80
```

10 Implemente el recurso `ingress-hello-test`.

```
kubectl apply -f ingress-hello-test.yaml
```

```
service/hello created
deployment.apps/hello created
```

11 Compruebe que la implementación de `hello` esté disponible.

```
kubectl get deployments
```

```
NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
hello                       3/3     3            3           4m59s
ingress-nginx-controller    1/1     1            1           3h39m
```

12 Obtenga la dirección IP pública del equilibrador de carga que utiliza la controladora de entrada de Nginx.

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      13m
```

13 Con un navegador, desplácese hasta la dirección IP pública y escriba la ruta de entrada.

```
http://10.19.14.76/hello
```

Se devolverá el mensaje "Hello".

```
{"message":"Hello"}
```

#### Resultados

El navegador permite acceder externamente a la aplicación de back-end que ofrece el servicio en ejecución dentro del clúster a través de la controladora de entrada con la dirección IP externa del equilibrador de carga.

## Entrada mediante Contour

Un recurso de entrada de Kubernetes proporciona enrutamiento HTTP o HTTPS desde fuera del clúster a uno o varios servicios dentro del clúster. Los clústeres de TKG admiten la entrada a través de controladoras de terceros, como Contour.

En este tutorial se demuestra cómo implementar la controladora de entrada de Contour para enrutar el tráfico externo a los servicios en un clúster de TKG. Contour es un proyecto de código abierto al que VMware contribuye.

#### Requisitos previos

- Revise el recurso de Entrada en la documentación de Kubernetes.
- Revise la controladora de entrada de Contour.
- Aprovisione un clúster de TKG.
- Conéctese al clúster de TKG.

#### Procedimiento

1 Cree un objeto de ClusterRoleBinding que permita a las cuentas de servicio administrar todos los recursos del clúster.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding
--clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

**Nota** Si se requiere una seguridad más estricta, use un objeto RoleBinding en el espacio de nombres de `projectcontour`. Consulte Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG.

2 Cree un espacio de nombres denominado `projectcontour`.

Este es el espacio de nombres predeterminado para la implementación de la controladora de entrada de Contour.

```
kubectl create ns projectcontour
```

3 Descargue el archivo YAML más reciente de la controladora de entrada de Contour: Implementación de entrada de Contour.

4 Abra el archivo `contour.yaml` con un editor de texto.

5 Comente las dos líneas siguientes anteponiendo el símbolo `#` a cada línea:

Línea 1632:

```
# service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp
```

Línea 1634:

```
# externalTrafficPolicy: Local
```

6 Implemente Contour mediante la aplicación del archivo `contour.yaml`.

```
kubectl apply -f contour.yaml
```

7 Compruebe que se hayan implementado el servicio de equilibrador de carga de Envoy y la controladora de entrada de Contour.

```
kubectl get services -n projectcontour
```

```
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
contour   ClusterIP      198.63.146.166  <none>          8001/TCP                     120m
envoy     LoadBalancer   198.48.52.47    192.168.123.5   80:30501/TCP,443:30173/TCP   120m
```

8 Verifique que los pods de Contour y Envoy estén en ejecución.

```
kubectl get pods -n projectcontour
```

```
NAME                        READY   STATUS      RESTARTS   AGE
contour-7966d6cdbf-skqfl    1/1     Running     1          21h
contour-7966d6cdbf-vc8c7    1/1     Running     1          21h
contour-certgen-77m2n       0/1     Completed   0          21h
envoy-fsltp                 1/1     Running     0          20h
```

9 Ejecute ping en el equilibrador de carga utilizando la dirección IP externa.

```
ping 192.168.123.5
```

```
PING 192.168.123.5 (192.168.123.5) 56(84) bytes of data.
64 bytes from 192.168.123.5: icmp_seq=1 ttl=62 time=3.50 ms
```

10 Cree un recurso de entrada denominado `ingress-nihao.yaml`.

Cree el archivo YAML.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-nihao
spec:
  rules:
  - http:
      paths:
      - path: /nihao
        backend:
          serviceName: nihao
          servicePort: 80
```

Aplique el archivo YAML.

```
kubectl apply -f ingress-nihao.yaml
```

Compruebe que se haya creado el recurso de entrada.

```
kubectl get ingress
```

El objeto de entrada utiliza la dirección IP externa para el equilibrador de carga Envoy (`192.168.123.5` en este ejemplo).

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-nihao   <none>   *       192.168.123.5     80      17s
```

11 Implemente un servicio de prueba con una aplicación de back-end.

Cree el siguiente archivo YAML con el nombre `ingress-nihao-test.yaml`.

```
kind: Service
apiVersion: v1
metadata:
  name: nihao
spec:
  selector:
    app: nihao
    tier: backend
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nihao
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nihao
      tier: backend
      track: stable
  template:
    metadata:
      labels:
        app: nihao
        tier: backend
        track: stable
    spec:
      containers:
        - name: nihao
          image: "gcr.io/google-samples/hello-go-gke:1.0"
          ports:
            - name: http
              containerPort: 80
```

Aplique el archivo YAML.

```
kubectl apply -f ingress-nihao-test.yaml
```

Compruebe que se haya creado el servicio `nihao`.

```
kubectl get services
```

```
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
nihao        ClusterIP   10.14.21.22     <none>        80/TCP     15s
```

Compruebe que se haya creado la implementación de back-end.

```
kubectl get deployments
```

```
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nihao   3/3     3            3           2m25s
```

Compruebe que existan los pods de back-end.

```
kubectl get pods
```

```
NAME                         READY   STATUS    RESTARTS   AGE
nihao-8646584495-9nm8x       1/1     Running   0          106s
nihao-8646584495-vscm5       1/1     Running   0          106s
nihao-8646584495-zcsdq       1/1     Running   0          106s
```

12 Obtenga la dirección IP pública del equilibrador de carga que utiliza la controladora de entrada de Contour.

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-nihao   <none>   *       10.19.14.76     80      13m
```

13 Con un navegador, desplácese hasta la dirección IP pública y escriba la ruta de entrada.

```
http://10.19.14.76/nihao
```

Se devolverá el mensaje "Hello".

```
{"message":"Hello"}
```

Resultados

El navegador permite acceder externamente a la aplicación de back-end que ofrece el servicio en ejecución dentro del clúster a través de la controladora de entrada con la dirección IP externa del equilibrador de carga.

# Usar clases de almacenamiento para volúmenes persistentes

Una directiva de almacenamiento de vSphere asignada a un espacio de nombres de vSphere genera dos ediciones de una clase de almacenamiento disponibles para su uso con volúmenes persistentes. La edición para elegir depende de los requisitos que usted establezca.

## Dos ediciones de una clase de almacenamiento disponible para su uso

Cuando se asigna una directiva de almacenamiento de vSphere a un espacio de nombres de vSphere, el sistema crea una clase de almacenamiento de Kubernetes coincidente en ese espacio de nombres de vSphere. Esta clase de almacenamiento se replica en cada clúster de TKG aprovisionado en ese espacio de nombres. A continuación, la clase de almacenamiento está disponible para crear volúmenes de almacenamiento persistentes para las cargas de trabajo de clúster.

Para cada directiva de almacenamiento de vSphere asignada a un espacio de nombres de vSphere, en Supervisor existe una sola clase de almacenamiento con el modo de enlace "Inmediato".

```
kubectl describe storageclass tkg-storage-policy
Name:                  tkg-storage-policy
IsDefaultClass:        No
Annotations:           cns.vmware.com/StoragePoolTypeHint=cns.vmware.com/vsan
Provisioner:           csi.vsphere.vmware.com
Parameters:            storagePolicyID=877b0f4b-b959-492a-b265-b4d460987b23
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>
```

Para cada clúster de TKG implementado en ese espacio de nombres de vSphere, hay dos clases de almacenamiento: una que es la misma que la clase de almacenamiento correspondiente en Supervisor y otra con `*-latebinding` anexado al nombre y "WaitForFirstConsumer" como modo de enlace.

```
kubectl get sc
NAME                              PROVISIONER              RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
tkg-storage-policy                csi.vsphere.vmware.com   Delete
Immediate              true                   2m43s
tkg-storage-policy-latebinding    csi.vsphere.vmware.com   Delete
WaitForFirstConsumer   true                   2m43s
```

Utilice la edición `latebinding` de la clase de almacenamiento cuando desee aprovisionar un volumen persistente después de que el programador de Kubernetes haya seleccionado el recurso informático. Para obtener más información, consulte Modo de enlace de volumen en la documentación de Kubernetes.

```
kubectl describe sc tkg-storage-policy
Name:                  tkg-storage-policy
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           csi.vsphere.vmware.com
Parameters:            svStorageClass=tkg-storage-policy
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>

kubectl describe sc tkg-storage-policy-latebinding
Name:                  tkg-storage-policy-latebinding
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           csi.vsphere.vmware.com
Parameters:            svStorageClass=tkg-storage-policy
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

## Aplicar revisiones a una clase de almacenamiento

Para TKG en Supervisor, no se puede crear una clase de almacenamiento manualmente mediante kubectl y YAML. Solo puede crear una clase de almacenamiento mediante el marco de directivas de almacenamiento de vSphere y aplicarla a un espacio de nombres de vSphere. El resultado es dos clases de almacenamiento correspondientes en cada clúster de TKG aprovisionado en ese espacio de nombres de vSphere.

Aunque no puede crear una clase de almacenamiento manualmente mediante kubectl y YAML, puede modificar una clase de almacenamiento existente mediante kubectl. Es posible que sea necesario hacerlo si aprovisionó un clúster de TKG sin especificar una clase de almacenamiento predeterminada y ahora desea implementar una aplicación mediante Helm o un paquete de Tanzu que requiere una clase de almacenamiento predeterminada.

En lugar de crear un clúster totalmente nuevo con almacenamiento predeterminado, puede aplicar revisiones a la clase de almacenamiento existente y agregar la anotación default = true, como se describe en la documentación de Kubernetes: Cambiar el objeto StorageClass predeterminado.

Por ejemplo: el siguiente comando devuelve dos clases de almacenamiento que están disponibles en el clúster de TKG:

```
kubectl describe sc
Name:                  gc-storage-profile
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           csi.vsphere.vmware.com
Parameters:            svStorageClass=gc-storage-profile
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>


Name:                  gc-storage-profile-latebinding
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           csi.vsphere.vmware.com
Parameters:            svStorageClass=gc-storage-profile
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                <none>
```

Utilice el siguiente comando para aplicar una revisión a una de las clases de almacenamiento y agregar la anotación:

```
kubectl patch storageclass gc-storage-profile -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
storageclass.storage.k8s.io/gc-storage-profile patched
```

Vuelva a comprobar las clases de almacenamiento y verá que una de ellas tiene revisiones para que ahora sea la predeterminada.

```
kubectl describe sc
Name:                  gc-storage-profile
IsDefaultClass:        Yes
Annotations:           storageclass.kubernetes.io/is-default-class=true
```

```
Provisioner:            csi.vsphere.vmware.com
Parameters:             svStorageClass=gc-storage-profile
AllowVolumeExpansion:   True
MountOptions:           <none>
ReclaimPolicy:          Delete
VolumeBindingMode:      Immediate
Events:                 <none>


Name:                   gc-storage-profile-latebinding
IsDefaultClass:         No
Annotations:            <none>
Provisioner:            csi.vsphere.vmware.com
Parameters:             svStorageClass=gc-storage-profile
AllowVolumeExpansion:   True
MountOptions:           <none>
ReclaimPolicy:          Delete
VolumeBindingMode:      WaitForFirstConsumer
Events:                 <none>
```

# Crear volúmenes de almacenamiento persistentes de forma dinámica

Es posible crear dinámicamente un volumen de almacenamiento persistente mediante una clase de almacenamiento existente y una notificación de volumen persistente (Persistent Volume Claim, PVC).

## PVC dinámica para clústeres de TKG

Para ejecutar cargas de trabajo con estado en clústeres de TKG, puede crear una notificación de volumen persistente para solicitar recursos de almacenamiento persistentes sin conocer los detalles de la infraestructura de almacenamiento subyacente. El almacenamiento que se emplea para la PVC se asigna a partir de la cuota de almacenamiento de espacio de nombres de vSphere.

La solicitud aprovisiona dinámicamente un objeto de volumen persistente y un disco virtual coincidente. La notificación está enlazada al volumen persistente. Cuando esta notificación se elimina, se eliminan también el objeto de volumen persistente y el disco virtual aprovisionado correspondientes.

Al crear la PVC, se crea de forma dinámica el volumen persistente de respaldo. La PVC hace referencia a la clase de almacenamiento **`tkg-store`**. La clase de almacenamiento se asocia con el espacio de nombres de vSphere en el que se aprovisionó el clúster de TKG de destino. Consulte Usar clases de almacenamiento para volúmenes persistentes para obtener más información.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tkg-cluster-pvc
```

```
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: tkg-store
  resources:
    requests:
        storage: 3Gi
```

Cree la PVC.

```
kubectl apply -f pvc_name.yaml
```

Compruebe la PVC.

```
kubectl get pvc my-pvc
```

Especifique la PVC en la especificación de implementación o de pod. Por ejemplo:

```
...
volumes:
    - name: my-pvc
      persistentVolumeClaim:
        claimName: my-pvc
```

# Crear volúmenes de almacenamiento persistentes de forma estática

Es posible crear estáticamente un volumen persistente (Persistent Volume, PV) en un clúster de TKG 2.0 con una notificación de volumen persistente (Persistent Volume Claim, PVC) de Supervisor.

## Definición de volumen persistente

A continuación, se muestra un ejemplo de definición de un volumen persistente (PV) estático. La definición requiere una clase de almacenamiento y un identificador de volumen. `volumeHandle` es el nombre de una notificación de volumen persistente (PVC) creada en el Supervisor en el mismo espacio de nombres de vSphere en el que se aprovisionó el clúster de TKG de destino. Esta PVC no debe asociarse a ningún pod.

Utilice el siguiente comando para obtener `storageClassName`.

```
kubectl get storageclass
```

Para `volumeHandle`, introduzca el nombre de la PVC en el Supervisor.

```
apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: static-tkg-block-pv
```

```
    annotations:
      pv.kubernetes.io/provisioned-by: csi.vsphere.vmware.com
  spec:
    storageClassName: gc-storage-profile
    capacity:
      storage: 2Gi
    accessModes:
      - ReadWriteOnce
    persistentVolumeReclaimPolicy: Delete
    claimRef:
      namespace: default
      name: static-tkg-block-pvc
    csi:
      driver: "csi.vsphere.vmware.com"
      volumeAttributes:
        type: "vSphere CNS Block Volume"
      volumeHandle: "supervisor-block-pvc-name"  #Enter the PVC name from Supervisor.
```

Utilice lo siguiente para crear el PV.

```
kubectl apply -f redis-leader-pvc.yaml -n guestbook
```

## Notificación de volumen persistente (PVC) para PV definido estáticamente

Si implementó el Supervisor en varias zonas.

Establezca la `storageClassName` en el mismo valor que en el PV.

```
kind: PersistentVolumeClaim
  apiVersion: v1
  metadata:
    name: static-tkg-block-pvc
  spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 2Gi
    storageClassName: gc-storage-profile
    volumeName: static-tkg-block-pv
```

Compruebe que la PVC esté enlazada al PV que creó.

```
kubectl get pv,pvc
```

# Implementar la aplicación del libro de visitas en un clúster de TKG

Implemente la aplicación del libro de visitas en el clúster de TKG y explore Kubernetes.

La implementación de la aplicación del libro de visitas es un método útil para explorar Kubernetes. La aplicación utiliza los objetos Deployment y ReplicaSet para implementar pods en el espacio de nombres predeterminado, y expone estos pods mediante Services. Los datos del libro de visitas se guardan para no perderlos si la aplicación deja de funcionar. En este tutorial, se utiliza una notificación de volumen persistente (Persistent Volume Claim, PVC) dinámica para solicitar recursos de almacenamiento persistentes sin conocer los detalles del almacenamiento subyacente. El almacenamiento que se emplea para la PVC se asigna a partir de la cuota de almacenamiento del espacio de nombres de vSphere. De forma predeterminada, los contenedores son efímeros y no tienen estado. Para las cargas de trabajo con estado, un método habitual consiste en crear una notificación de volumen persistente (Persistent Volume Claim, PVC). Puede utilizar una PVC para montar los volúmenes persistentes y acceder al almacenamiento. La solicitud aprovisiona dinámicamente un objeto de volumen persistente y un disco virtual coincidente. La notificación está enlazada al volumen persistente. Cuando esta notificación se elimina, se eliminan también el objeto de volumen persistente y el disco virtual aprovisionado correspondientes.

#### Requisitos previos

Revise los siguientes temas:

- Tutorial de la aplicación del libro de visitas en la documentación de Kubernetes
- Aprovisione un clúster de TKG.
- Conéctese al clúster de TKG.

#### Procedimiento

1 Inicie sesión en el clúster de TKG.

2 Cree el espacio de nombres del libro de visitas.

```
kubectl create namespace guestbook
```

Compruebe lo siguiente:

```
kubectl get ns
```

3 Cree un control de acceso basado en funciones mediante la instancia de PSP con privilegios predeterminada.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

**Nota** Si se requiere más seguridad, aplique un objeto RoleBinding en el espacio de nombres del libro de visitas.

4 Compruebe la clase de almacenamiento o cree una.

Para comprobar una clase de almacenamiento existente:

```
kubectl describe namespace
```

O bien, si tiene permisos de administrador de vSphere:

```
kubectl get storageclass
```

5 Cree los archivos YAML de las notificaciones de volumen persistente dinámicas que hacen referencia a la clase de almacenamiento.

Utilice los siguientes archivos YAML. Actualice cada uno con el nombre de la clase de almacenamiento.

- Notificación de volumen persistente guía de Redis
- Notificación de volumen persistente de seguimiento de Redis

6 Aplique las PVC del libro de visitas al clúster.

```
kubectl apply -f redis-leader-pvc.yaml -n guestbook
```

```
kubectl apply -f redis-follower-pvc.yaml -n guestbook
```

7 Compruebe el estado de las PVC.

```
kubectl get pvc,pv -n guestbook
```

Las PVC y los volúmenes persistentes (persistent volumes, PV) se enumeran y están disponibles para su uso.

```
NAME                                          STATUS
VOLUME                                        CAPACITY   ACCESS MODES   STORAGECLASS
AGE
persistentvolumeclaim/redis-follower-pvc   Bound    pvc-37b72f35-3de2-4f84-
be7d-50d5dd968f62   2Gi        RWO            tkgs-storage-class   66s
persistentvolumeclaim/redis-leader-pvc     Bound    pvc-2ef51f31-dd4b-4fe2-bf4c-
f0149cb4f3da   2Gi        RWO            tkgs-storage-class   66s

NAME                                                   CAPACITY   ACCESS MODES   RECLAIM
POLICY STATUS   CLAIM                  STORAGECLASS
persistentvolume/pvc-2ef51f31-dd4b-4fe2-bf4c   2Gi   RWO            Delete
Bound    guestbook/redis-leader-pvc     tkgs-storage-class
persistentvolume/pvc-37b72f35-3de2-4f84-be7d   2Gi   RWO            Delete
Bound    guestbook/redis-follower-pvc   tkgs-storage-class
```

8 Cree los archivos YAML del libro de visitas.

Usar los siguientes archivos YAML:

- Implementación guía de Redis

- Implementación de seguimiento de Redis
- Servicio guía de Redis
- Servicio de seguimiento de Redis
- Implementación de front-end del libro de visitas
- Servicio front-end del libro de visitas

9 Implemente la aplicación del libro de visitas en su espacio de nombres.

```
kubectl apply -f . --namespace guestbook
```

10 Compruebe la creación de los recursos del libro de visitas.

```
kubectl get all -n guestbook
```

```
NAME                                                    READY   STATUS
RESTARTS   AGE
pod/guestbook-frontend-deployment-56fc5b6b47-cd58r    1/1     Running
0          65s
pod/guestbook-frontend-deployment-56fc5b6b47-fh6dp    1/1     Running
0          65s
pod/guestbook-frontend-deployment-56fc5b6b47-hgd2b    1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-99fgw        1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-rhxf7        1/1     Running
0          65s
pod/redis-leader-deployment-7d89bbdbcf-flt4q          1/1     Running
0          65s

NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend           LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower               ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                 ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

11 Acceda a la página web del libro de visitas mediante la dirección `External-IP` del equilibrador de carga de `service/guestbook-frontend` que, en este ejemplo, es `10.19.15.99`.

Puede ver la interfaz web del libro de visitas y puede introducir valores en la base de datos de dicho libro. Si reinicia la aplicación, los datos se conservan.

# YAML de la aplicación del libro de visitas

Utilice el ejemplo de archivos YAML para implementar la aplicación del libro de visitas con almacenamiento de datos persistentes.

## Notificación de volumen persistente guía de Redis

El archivo `redis-leader-pvc.yaml` es un ejemplo de notificación de volumen persistente que hace referencia a una clase de almacenamiento con nombre. Para utilizar este ejemplo, introduzca el nombre de la clase de almacenamiento.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-leader-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkg-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

## Notificación de volumen persistente de seguimiento de Redis

El archivo `redis-follower-pvc.yaml` es un ejemplo de notificación de volumen persistente que hace referencia a una clase de almacenamiento con nombre. Para utilizar este ejemplo, introduzca el nombre de la clase de almacenamiento.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-follower-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkg-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

## Implementación guía de Redis

El archivo `redis-leader-deployment.yaml` es un ejemplo de implementación guía de Redis con un volumen persistente.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-leader-deployment
spec:
  selector:
    matchLabels:
      app: redis
      role: leader
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: leader
        tier: backend
    spec:
      containers:
      - name: leader
        image: redis:6.0.5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-leader-data
          mountPath: /data
      volumes:
      - name: redis-leader-data
        persistentVolumeClaim:
          claimName: redis-leader-pvc
```

## Implementación de seguimiento de Redis

El archivo `redis-follower-deployment.yaml` es un ejemplo de implementación de seguimiento de Redis con un volumen persistente.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-follower-deployment
  labels:
    app: redis
spec:
  selector:
```

```
    matchLabels:
      app: redis
      role: follower
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: follower
        tier: backend
    spec:
      containers:
      - name: follower
        image: gcr.io/google_samples/gb-redis-follower:v2
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-follower-data
          mountPath: /data
      volumes:
      - name: redis-follower-data
        persistentVolumeClaim:
          claimName: redis-follower-pvc
```

## Servicio guía de Redis

El archivo `redis-leader-service.yaml` es un ejemplo de servicio guía de Redis.

```
apiVersion: v1
kind: Service
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    role: leader
    tier: backend
```

## Servicio de seguimiento de Redis

El archivo `redis-follower-service.yaml` es un ejemplo de servicio de seguimiento de Redis.

```
apiVersion: v1
kind: Service
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  ports:
  - port: 6379
  selector:
    app: redis
    role: follower
    tier: backend
```

## Implementación de front-end del libro de visitas

El archivo `guestbook-frontend-deployment.yaml` es un ejemplo de implementación de front-end de un libro de visitas.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-frontend-deployment
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

## Servicio front-end del libro de visitas

El archivo `guestbook-frontend-service.yaml` es un ejemplo de servicio de equilibrador de carga de front-end de un libro de visitas.

```
apiVersion: v1
kind: Service
metadata:
  name: guestbook-frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: guestbook
    tier: frontend
```

# Implementar la aplicación StatefulSet en Zonas de vSphere con asociación de volúmenes de enlace en tiempo de ejecución

En este ejemplo, se demuestra cómo implementar una aplicación StatefulSet en un clúster de TKG en el Supervisor.

## Clase de almacenamiento

Es posible elegir dos ediciones de la clase de almacenamiento. Para esta implementación, utilizaremos la edición `*-latebinding`.

```
kubectl get sc
NAME                                PROVISIONER              RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
zonal-ds-policy-105                 csi.vsphere.vmware.com   Delete
Immediate              true                   17h
zonal-ds-policy-105-latebinding   csi.vsphere.vmware.com   Delete
WaitForFirstConsumer   true                   17h
```

## Topología de Supervisor de varias zonas

El clúster de TKG se aprovisiona en el Supervisor en Zonas de vSphere.

```
kubectl get nodes -L topology.kubernetes.io/zone
NAME                                                               STATUS
ROLES                  AGE   VERSION            ZONE
test-cluster-e2e-script-105-m72sb-2dnsz                           Ready    control-
plane,master   18h   v1.22.6+vmware.1   zone-1
test-cluster-e2e-script-105-m72sb-rmtjn                           Ready    control-
```

```
plane,master   18h   v1.22.6+vmware.1   zone-2
test-cluster-e2e-script-105-m72sb-rvhb8                        Ready    control-
plane,master   18h   v1.22.6+vmware.1   zone-3
test-cluster-e2e-script-105-nodepool-1-p86fm-6dfcdc77b7-fxm4s   Ready
<none>                 18h   v1.22.6+vmware.1   zone-1
test-cluster-e2e-script-105-nodepool-2-gx5gs-7cf4895b77-6wlb4   Ready
<none>                 18h   v1.22.6+vmware.1   zone-2
test-cluster-e2e-script-105-nodepool-3-fkkc9-856cd45985-d8nsl   Ready
<none>                 18h   v1.22.6+vmware.1   zone-3
```

## StatefulSet

StatefulSet (`sts.yaml`) implementa la aplicación en los pods, cada uno con un identificador persistente que se mantiene en todas las reprogramaciones.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  serviceName: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                - zone-1
                - zone-2
                - zone-3
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - nginx
            topologyKey: topology.kubernetes.io/zone
      containers:
        - name: nginx
          image: gcr.io/google_containers/nginx-slim:0.8
```

```
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
            - name: logs
              mountPath: /logs
  volumeClaimTemplates:
    - metadata:
        name: www
      spec:
        accessModes: [ "ReadWriteOnce" ]
        storageClassName: zonal-ds-policy-105-latebinding
        resources:
          requests:
            storage: 2Gi
    - metadata:
        name: logs
      spec:
        accessModes: [ "ReadWriteOnce" ]
        storageClassName: zonal-ds-policy-105-latebinding
        resources:
          requests:
            storage: 1Gi
```

## Implementar la aplicación

Implemente la aplicación StatefulSet de la siguiente manera. Con una implementación correcta, los pods de la aplicación se programan en 3 Zonas de vSphere y se espera al primer consumidor que utiliza la clase de almacenamiento con el modo de volumen de enlace en tiempo de ejecución.

1 Implemente StatefulSet.

```
kubectl create -f sts.yaml
statefulset.apps/web created
```

2 Verifique StatefulSet.

```
kubectl get statefulset
NAME   READY   AGE
web    3/3     112s
```

3 Compruebe los pods.

```
kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE    IP
NODE                                                            NOMINATED NODE   READINESS
GATES
web-0   1/1     Running   0          117s   172.16.1.2   test-cluster-e2e-script-105-
nodepool-3-fkkc9-856cd45985-d8nsl   <none>           <none>
```

```
web-1   1/1     Running   0          90s    172.16.2.2   test-cluster-e2e-script-105-
nodepool-2-gx5gs-7cf4895b77-6wlb4   <none>           <none>
web-2   1/1     Running   0          53s    172.16.3.2   test-cluster-e2e-script-105-
nodepool-1-p86fm-6dfcdc77b7-fxm4s   <none>           <none>
```

4 Compruebe la programación de pods en las zonas y el enlace de volúmenes en tiempo de ejecución.

```
kubectl get pv -o=jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.spec.claimRef.name}
{"\t"}{.spec.nodeAffinity}{"\n"}{end}'
pvc-7010597f-31cf-4ab1-bbd7-98ac04e0c603    www-
web-2    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-1"]}]}]}}
pvc-921fadfc-df89-456d-a341-00f4117035f8    logs-
web-0    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-3"]}]}]}}
pvc-bcb46a24-58cb-4ec7-a964-391fe80400fc    www-
web-1    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-2"]}]}]}}
pvc-f51a44e5-ec19-4bec-b67a-3e34512049b8    www-
web-0    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-3"]}]}]}}
pvc-fa68887a-31dd-4d9e-bb39-88653a9d80c9    logs-
web-2    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-1"]}]}]}}
pvc-fc2cd6f7-b033-48ee-892d-df5318ec6f3e    logs-
web-1    {"required":{"nodeSelectorTerms":[{"matchExpressions":
[{"key":"topology.kubernetes.io/zone","operator":"In","values":["zone-2"]}]}]}}
```

# Implementar cargas de trabajo de AI/ML en clústeres de Servicio TKG

13

Las cargas de trabajo de AI/ML se pueden implementar en clústeres de Servicio TKG mediante vGPU de NVIDIA. La implementación de cargas de trabajo de AI/ML requiere una configuración inicial por parte del administrador de vSphere y una configuración por parte del operador del clúster. Una vez que el entorno está habilitado para vGPU, los desarrolladores pueden implementar cargas de trabajo de AI/ML basadas en contenedores en los clústeres de Servicio TKG.

Lea los siguientes temas a continuación:



## Acerca de la implementación de cargas de trabajo de AI/ML en clústeres de Servicio TKG

Las cargas de trabajo de AI/ML se pueden implementar en clústeres de Servicio TKG mediante la tecnología GPU de NVIDIA.

### Compatibilidad con TKGS para cargas de trabajo de AI/ML

Es posible implementar cargas de trabajo de uso intensivo de recursos informáticos en clústeres de Servicio TKG. En este contexto, una carga de trabajo con uso intensivo de recursos informáticos es una aplicación de inteligencia artificial (AI) o aprendizaje automático (ML) que requiere el uso de un dispositivo acelerador de GPU.

Para facilitar la ejecución de cargas de trabajo de AI/ML en un entorno de Kubernetes, VMware se asoció con NVIDIA para admitir la plataforma GPU Cloud de NVIDIA. Esto significa que puede implementar imágenes de contenedor desde el catálogo de NGC en clústeres de TKGS. Para obtener más información sobre la compatibilidad con GPU de NVIDIA en vSphere 8, consulte el artículo de vGPU en la zona técnica.

## Modos GPU compatibles

La implementación de cargas de trabajo de AI/ML basadas en NVIDIA en clústeres de Servicio TKG requiere el uso de la edición para Ubuntu de Tanzu Kubernetes, versiones 1.22 o posteriores. vSphere admite dos modos: vGPU de NVIDIA Grid y Acceso directo de GPU con un dispositivo de DirectPath I/O dinámico. Para obtener más información, consulte Sistemas operativos compatibles y plataformas de Kubernetes en la documentación de NVIDIA.

### Tabla 13-1. Máquinas virtuales de vSphere con vGPU de NVIDIA

| Sistema operativo | TKr | vSphere with Tanzu | Descripción |
| --- | --- | --- | --- |
| Ubuntu 20.04 LTS | 1.22-1.2x* (última hasta 1.28) | 7.0 U3c<br>8.0 U2 o superior | El controlador del administrador de hosts NVIDIA instalado en cada host ESXi virtualiza el dispositivo GPU. Este se comparte después entre varias GPU virtuales (vGPU) de NVIDIA.<br>**Nota** El planificador de recursos distribuidos (Distributed Resource Scheduler, DRS) de vSphere distribuye las máquinas virtuales de vGPU usando el método “breadth-first” entre los hosts que componen un clúster de vSphere. Para obtener más información, consulte Colocación de DRS de máquinas virtuales de vGPU en la guía Administrar recursos de vSphere.<br>Cada vGPU de NVIDIA se define por la cantidad de memoria del dispositivo GPU. Por ejemplo, si el dispositivo GPU tiene una cantidad total de 32 GB de RAM, puede crear 8 vGPU con 4 GB de memoria cada uno. |

### Tabla 13-2. Máquinas virtuales de vSphere con Acceso directo de GPU

| Sistema operativo | TKr | vSphere with Tanzu | Descripción |
| --- | --- | --- | --- |
| Ubuntu 20.04 LTS | 1.22-1.2x* (última hasta 1.28) | 7.0 U3c<br>8.0 U2 o superior | En la misma clase de máquina virtual en la que se configura el perfil de la vGPU de NVIDIA, se incluye compatibilidad con un dispositivo de redes de acceso directo mediante la instancia dinámica de DirectPath I/O. En este caso, vSphere DRS determina la colocación de máquinas virtuales. |

# Flujo de trabajo del administrador de vSphere para implementar cargas de trabajo de AI/ML en clústeres TKGS

Para permitir que los desarrolladores implementen cargas de trabajo de AI/ML en clústeres TKG, como administrador de vSphere, configure el entorno de Supervisor para que admita el hardware de NVIDIA GPU.

## Paso 1 del administrador: revisar los requisitos del sistema

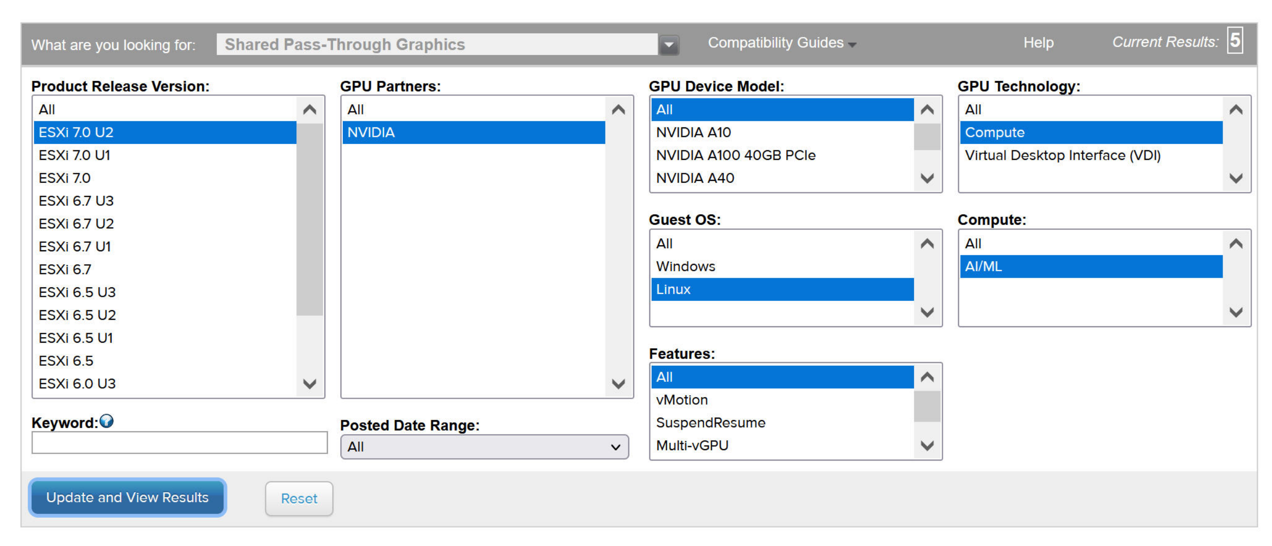
Consulte los siguientes requisitos del sistema para configurar el entorno de implementación de cargas de trabajo de AI/ML en clústeres de TKG.

| Requisito | Descripción |
| --- | --- |
| infraestructura de vSphere 8 | vCenter Server y hosts ESXi |
| Licencia de administración de cargas de trabajo | Espacios de nombres de vSphere y Supervisor |
| OVA de TKR Ubuntu | Notas de la versión de las versiones de Tanzu Kubernetes |
| Controlador de host NVIDIA vGPU | Descargue el VIB del sitio web de NGC. Si desea más información, consulte la documentación del controlador del software vGPU. |
| Servidor de licencias NVIDIA para vGPU | FQDN proporcionado por la organización |

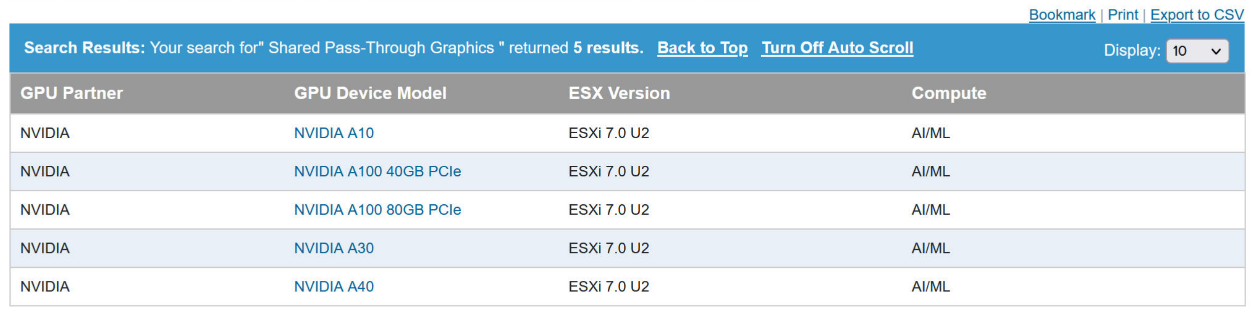
## Paso 2 del administrador: instalar un dispositivo NVIDIA GPU compatible en hosts ESXi

Para implementar cargas de trabajo de AI/ML en TKG, instale uno o varios dispositivos NVIDIA GPU compatibles en cada host ESXi que contenga el clúster de vCenter en el que se habilitará **Administración de cargas de trabajo**.

Para ver los dispositivos NVIDIA GPU compatibles, consulte la guía de compatibilidad de VMware.




**Click on the 'GPU Device Model' to view more details and to subscribe to RSS feeds.**

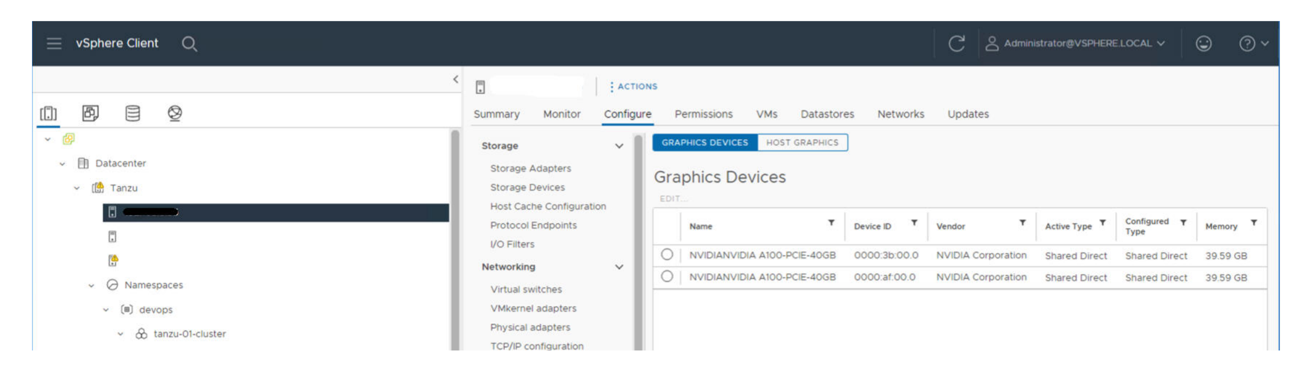

Bookmark | Print | Export to CSV

**Search Results:** Your search for" Shared Pass-Through Graphics " returned **5 results.** Back to Top Turn Off Auto Scroll Display: 10

| GPU Partner | GPU Device Model | ESX Version | Compute |
|---|---|---|---|
| NVIDIA | NVIDIA A10 | ESXi 7.0 U2 | AI/ML |
| NVIDIA | NVIDIA A100 40GB PCIe | ESXi 7.0 U2 | AI/ML |
| NVIDIA | NVIDIA A100 80GB PCIe | ESXi 7.0 U2 | AI/ML |
| NVIDIA | NVIDIA A30 | ESXi 7.0 U2 | AI/ML |
| NVIDIA | NVIDIA A40 | ESXi 7.0 U2 | AI/ML |

El dispositivo NVIDIA GPU debe admitir los perfiles de vGPU NVIDIA AI Enterprise (NVAIE) más recientes. Consulte la documentación de GPU compatibles con el software NVIDIA Virtual GPU para obtener instrucciones.

Por ejemplo, el siguiente host ESXi tiene dos dispositivos NVIDIA GPU A100 instalados.




# Paso 3 del administrador: configurar cada host ESXi para operaciones de vGPU

Para cada host ESXi que contenga el clúster de vCenter en el que esté habilitada la **Administración de cargas de trabajo**, configure el host para NVIDIA vGPU habilitando Compartidos directos y SR-IOV.

### Habilitar Compartidos directos en cada host ESXi

Para aprovechar la funcionalidad NVIDIA vGPU, habilite el modo **Compartidos directos** en cada host ESXi que contenga el clúster de vCenter en el que esté habilitada la **Administración de cargas de trabajo**.

Para habilitar **Compartidos directos**, realice los siguientes pasos. Para obtener más instrucciones, consulte Configurar gráficos virtuales en vSphere.

1 Inicie sesión en vCenter Server mediante vSphere Client.

2 Seleccione un host ESXi en el clúster de vCenter.

3 Seleccione **Configurar > Hardware > Gráficos > Dispositivos de gráficos**.

4 Seleccione el dispositivo acelerador de NVIDIA GPU.

5 **Edite** la configuración de dispositivos de gráficos.

6 Seleccione **Compartidos directos**.

7 Para la **Directiva de asignación de GPU de acceso directo compartido**, si desea obtener el mejor rendimiento, seleccione **Máquinas virtuales extendidas entre las GPU**

8 Haga clic en **Aceptar** para guardar la configuración.

9 Tenga en cuenta que la configuración se aplicará después de reiniciar el host.

10 Haga clic con el botón secundario en el host ESXi y póngalo en el modo de mantenimiento.

11 Reinicie el host.

12 Cuando el host vuelva a ejecutarse, salga del modo de mantenimiento.

13 Repita este proceso para cada host ESXi en el clúster de vSphere que admita la **Administración de cargas de trabajo**.

### Activar el BIOS de SR-IOV para dispositivos NVIDIA GPU A30 y A100

Si utiliza los dispositivos GPU NVIDIA A30 o A100, los cuales son necesarios para GPU de varias instancias (modo MIG), debe habilitar SR-IOV en el host ESXi. Si SR-IOV no está habilitado, no se pueden iniciar las máquinas virtuales del nodo del clúster de Tanzu Kubernetes. Si esto ocurre, verá el siguiente mensaje de error en el panel **Tareas recientes** de vCenter Server en el que está habilitada **Administración de cargas de trabajo**.

```
Could not initialize plugin libnvidia-vgx.so for vGPU nvidia_aXXX-xx. Failed to start
the virtual machine. Module DevicePowerOn power on failed.
```

Para habilitar SR-IOV, inicie sesión en el host ESXi mediante la consola web. Seleccione **Administrar > Hardware** . Seleccione el dispositivo NVIDIA GPU y haga clic en **Configurar SR-IOV**. Desde ahí, puede activar SR-IOV. Para ver más instrucciones, consulte Single Root I/O Virtualization (SR-IOV) en la documentación de vSphere.

### vGPU con Instancia dinámica de DirectPath I/O (Dispositivo habilitado para acceso directo)

Si utiliza vGPU con Instancia dinámica de DirectPath I/O, realice la siguiente configuración adicional.

1 Inicie sesión en vCenter Server mediante vSphere Client.
2 Seleccione el host ESXi de destino en el clúster de vCenter.
3 Seleccione **Configurar > Hardware > Dispositivos PCI**.
4 Seleccione la pestaña **Todos los dispositivos PCI**.
5 Seleccione el dispositivo acelerador de NVIDIA GPU de destino.
6 Haga clic en **Alternar acceso directo**.
7 Haga clic con el botón secundario en el host ESXi y póngalo en el modo de mantenimiento.
8 Reinicie el host.
9 Cuando el host vuelva a ejecutarse, salga del modo de mantenimiento.

## Paso 4 del administrador: instalar el controlador del administrador de hosts de NVIDIA en cada host ESXi

Para ejecutar las máquinas virtuales del nodo del clúster de Tanzu Kubernetes con aceleración de gráficos NVIDIA vGPU, instale el controlador del administrador de hosts de NVIDIA en cada host ESXi que contenga el clúster de vCenter en el que se habilitará **Administración de cargas de trabajo**.

Los componentes del controlador del administrador de hosts NVIDIA vGPU se empaquetan en un paquete de instalación de vSphere (VIB). La organización le proporciona el VIB de NVAIE a través de su programa de licencias NVIDIA GRID. VMware no proporciona los VIB de NVAIE ni hace que estén disponibles para descargarlos. Como parte del programa de licencias NVIDIA, su organización configura un servidor de licencias. Consulte la Guía de inicio rápido del software de GPU virtual para obtener más información.

Una vez que se configure el entorno de NVIDIA, ejecute el siguiente comando en cada host ESXi, reemplace la dirección del servidor de licencias NVIDIA y la versión del VIB de NVAIE con los valores adecuados para su entorno. Para obtener más instrucciones, consulte Instalar y configurar el VIB de NVIDIA en ESXi en la base de conocimientos de soporte de VMware.

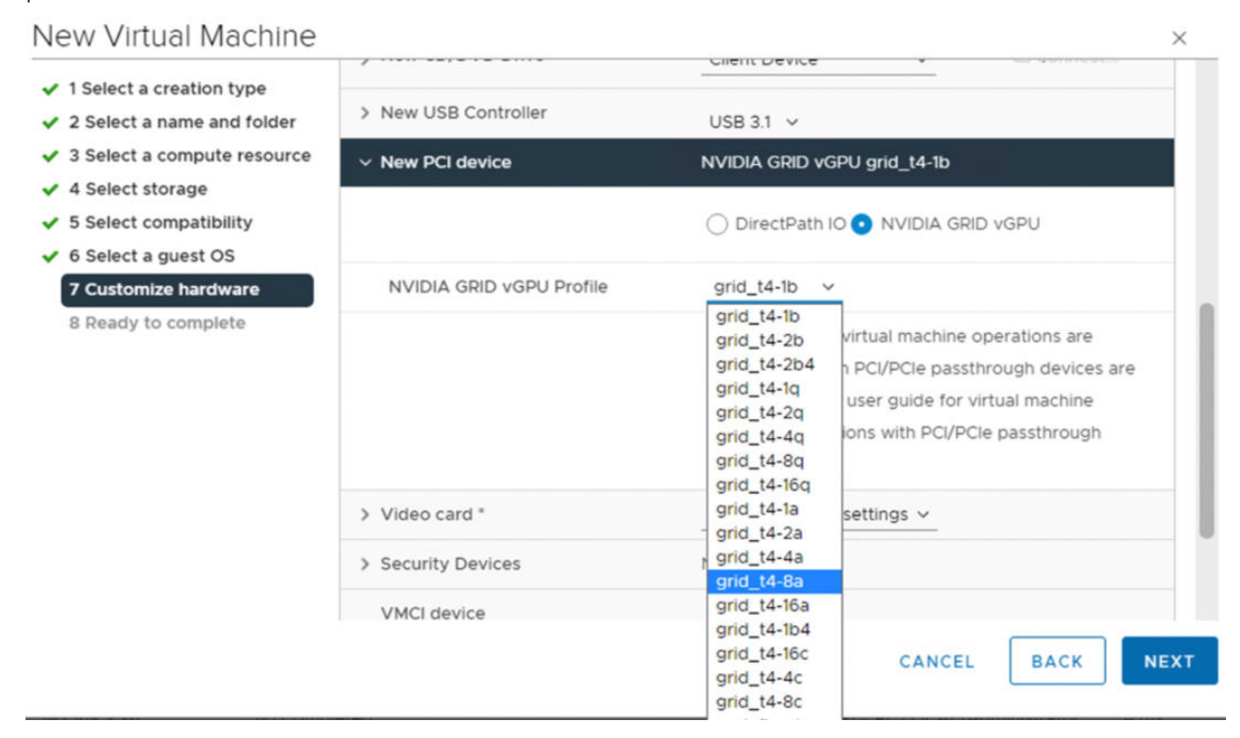
**Nota** La versión del VIB de NVAIE instalada en los hosts ESXi debe coincidir con la versión de software de vGPU instalada en las máquinas virtuales del nodo. La siguiente versión es solo un ejemplo.

```
esxcli system maintenanceMode set --enable true
esxcli software vib install -v ftp://server.domain.example.com/nvidia/signed/
NVIDIA_bootbank_NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15525992.vib
esxcli system maintenanceMode set --enable false
/etc/init.d/xorg restart
```

# Paso 5 del administrador: comprobar que los hosts ESXi estén listos para las operaciones de NVIDIA vGPU

Para comprobar que cada host ESXi esté listo para las operaciones de NVIDIA vGPU, realice las siguientes comprobaciones en cada host ESXi del clúster de vCenter en el que se habilitará **Administración de cargas de trabajo**:

- Acceda mediante SSH al host ESXi, entre en el modo de shell y ejecute el comando `nvidia-smi`. La interfaz de administración del sistema NVIDIA es una utilidad de línea de comandos que proporciona el administrador de hosts de NVIDIA vGPU. Al ejecutar este comando, se devuelven los controladores y las GPU en el host.
- Ejecute el siguiente comando para comprobar que el controlador de NVIDIA esté instalado correctamente: `esxcli software vib list | grep NVIDA`.
- Compruebe que el host esté configurado con Compartidos directos de GPU y que SR-IOV esté activado (si utiliza dispositivos NVIDIA A30 o A100).
- Con vSphere Client, en el host ESXi que está configurado para GPU, cree una nueva máquina virtual con un dispositivo PCI incluido. El perfil de NVIDIA vGPU debe aparecer y se debe poder seleccionar.



## Paso 6 del administrador: habilitar administración de cargas de trabajo

Para habilitar **Administración de cargas de trabajo**, consulte Implementar clústeres de Servicio TKG.

**Nota** Omita este paso si ya tiene un clúster de vSphere con la **Administración de cargas de trabajo** habilitada, siempre que ese clúster utilice los hosts ESXi que configuró para vGPU.

## Paso 7 del administrador: crear o actualizar una biblioteca de contenido con una TKR para Ubuntu

NVIDIA vGPU requiere el sistema operativo Ubuntu. No es posible utilizar la edición de PhotonOS de una versión de Tanzu Kubernetes para clústeres de vGPU.

VMware proporciona ediciones de Ubuntu de versiones de Tanzu Kubernetes. A partir de vSphere 8, la edición de Ubuntu se especifica mediante una anotación en el YAML del clúster.

Cree o actualice una biblioteca de contenido existente con una TKR de Ubuntu compatible. Consulte Capítulo 5 Administrar las versiones de Kubernetes para clústeres de Servicio TKG.

**Nota** Omita este paso si ya tiene una biblioteca de contenido TKR existente configurada en vCenter. No cree una segunda biblioteca de contenido para las TKR. Si lo hace, puede provocar inestabilidad en el sistema.

## Paso 8 del administrador: crear una clase de máquina virtual personalizada con el perfil de vGPU

Cree una clase de máquina virtual personalizada con un perfil de vGPU. A continuación, utilizará esta clase de máquina virtual en la especificación del clúster para crear los nodos del clúster de TKGS. Consulte las siguientes instrucciones: Crear una clase de máquina virtual personalizada para dispositivos vGPU de NVIDIA.

## Paso 9 del administrador: Configurar el espacio de nombres de vSphere

Cree un espacio de nombres de vSphere para cada clúster de TKG vGPU que tenga previsto aprovisionar. Consulte Crear un espacio de nombres de vSphere para alojar clústeres de Servicio TKG.

Para configurar el espacio de nombres de vSphere, agregue usuarios o grupos de SSO de vSphere con permisos de edición y asocie una directiva de almacenamiento para volúmenes persistentes. Consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

Asocie la biblioteca de contenido de TKR en la que se almacena la imagen de Ubuntu deseada con el espacio de nombres de vSphere. Consulte Asociar la biblioteca de contenido de TKR al Servicio TKG.

Asocie la clase de máquina virtual personalizada con el espacio de nombres de vSphere.

- En el espacio de nombres de vSphere, seleccione el mosaico **Servicios de máquina virtual** y haga clic en **Administrar clases de máquina virtual**.
- Busque la clase de máquina virtual personalizada que creó en la lista de clases.
- Seleccione la clase y haga clic en **Agregar**.

Para obtener más instrucciones, consulte Asociar las clases de máquinas virtuales con el espacio de nombres de vSphere.

## Paso 10 del administrador: Comprobar que Supervisor esté listo

La última tarea de administración consiste en comprobar que Supervisor esté aprovisionado y disponible para que lo pueda utilizar el operador del clúster a fin de aprovisionar un clúster de TKG para cargas de trabajo de AI/ML.

Consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

# Flujo de trabajo de operadores de clúster para implementar cargas de trabajo de AI/ML en clústeres de servicio TKG

Para permitir que los desarrolladores implementen cargas de trabajo de AI/ML en clústeres de servicio TKG, como operador de clúster, cree uno o varios clústeres de Kubernetes e instale los operadores de red y GPU de NVIDIA en cada uno.

## Paso 1 del operador: verificar los requisitos previos

En estas instrucciones se supone que el administrador de vSphere configuró el entorno para GPU de NVIDIA. Consulte Flujo de trabajo del administrador de vSphere para implementar cargas de trabajo de AI/ML en clústeres TKGS.

En estas instrucciones se supone que va a instalar la edición NVIDIA AI Enterprise (NVAIE) del operador de GPU, la cual está preconfigurada y optimizada para usarla con vSphere IaaS control plane. El operador de GPU de NVAIE es diferente del operador de GPU que está disponible en el catálogo público de NGC. Consulte NVIDIA AI Enterprise para obtener más información.

En estas instrucciones se supone que está utilizando una versión del operador de GPU NVAIE y el controlador de vGPU que tiene un VIB correspondiente para ESXi. Consulte Versiones del operador de GPU de NVIDIA para obtener más información.

Al aprovisionar el clúster de TKG, debe utilizar la edición de Ubuntu de la TKR. Con TKG en Supervisor de vSphere 8, la edición de Ubuntu se especifica en el YAML del clúster mediante la anotación.

# Paso 2 del operador: Aprovisionar un clúster de TKGS para vGPU de NVIDIA

VMware ofrece compatibilidad nativa con TKGS para GPU virtuales de NVIDIA en servidores certificados de GPU de NVIDIA con Operador de GPU de NVIDIA y Operador de redes de NVIDIA. Estos operadores se instalan en un clúster de carga de trabajo de TKGS. Si desea aprovisionar un clúster de TKGS para que aloje cargas de trabajo de vGPU, realice los siguientes pasos.

1 Instale Herramientas de la CLI de Kubernetes para vSphere.

   Consulte Instalar el Herramientas de la CLI de Kubernetes para vSphere.

2 Si utiliza el complemento de vSphere para kubectl, autentíquese en el Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS-or-FQDN --vsphere-username USERNAME
```

   **Nota** El FQDN solo se puede utilizar si se habilitó Supervisor con él.

3 Con kubectl, cambie el contexto al espacio de nombres de vSphere que creó el administrador de vSphere para el clúster de vGPU de TKGS.

```
kubectl config get-contexts
```

```
kubectl config use-context TKG-GPU-CLUSTER-NAMESPACE
```

4 Obtenga el nombre de la clase de máquina virtual personalizada con el perfil de vGPU que creó el administrador de vSphere.

```
kubectl get virtualmachineclass
```

   **Nota** La clase de máquina virtual debe estar enlazada al espacio de nombres de vSphere de destino.

5 Obtenga el TKR NAME de la versión de Tanzu Kubernetes de Ubuntu que sincronizó el administrador de vSphere desde la biblioteca de contenido y agregó al espacio de nombres de vSphere.

```
kubectl get tkr
```

6 Cree el YAML para aprovisionar el clúster de TKG habilitado para vGPU.

   a Decida qué API de aprovisionamiento de clústeres de TKGS va a utilizar, API v1alpha3 o API v1beta1: API de aprovisionamiento de clústeres de TKG.

   b Según la API que elija, consulte el ejemplo de Ubuntu correspondiente a esa API.

      - Ejemplo de v1alpha3: TKC con Ubuntu TKR

- Ejemplo de v1beta1: clúster con Ubuntu TKR

**Nota** Debe utilizar una imagen de sistema operativo Ubuntu. No puede utilizar Photon OS.

c Utilice la información que recopiló de los resultados de los comandos anteriores para personalizar la especificación del clúster de TKGS.

7 Ejecute el siguiente comando kubectl para aprovisionar el clúster.

```
kubectl apply -f CLUSTER-NAME.yaml
```

Por ejemplo:

```
kubectl apply -f tkg-gpu-cluster-1.yaml
```

8 Compruebe el aprovisionamiento del clúster.

Supervise la implementación de nodos del clúster mediante kubectl.

```
kubectl get tanzukubernetesclusters -n NAMESPACE
```

9 Inicie sesión en el clúster de vGPU de TKGS con complemento de vSphere para kubectl.

```
kubectl vsphere login --server=IP-ADDRESS-or-FQDN --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace
NAMESPACE-NAME
```

10 Compruebe el clúster.

Con los comandos siguientes, compruebe el clúster:

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

## Paso 3 del operador: Instalar el operador de redes de NVIDIA

El operador de redes de NVIDIA aprovecha los recursos personalizados de Kubernetes y el marco del operador a fin de optimizar las redes para vGPU. Si desea obtener más información, consulte Operador de redes de NVIDIA.

1 Compruebe que haya iniciado sesión en el clúster de carga de trabajo de vGPU de TKGS y que el contexto esté establecido en el espacio de nombres del clúster de carga de trabajo correspondiente a la vGPU de TKGS.

Consulte las instrucciones Paso 2 del operador: Aprovisionar un clúster de TKGS para vGPU de NVIDIA si es necesario.

2 Para instalar Helm, consulte la documentación de Helm.

3 Recupere el gráfico de Helm del operador de redes de NVIDIA.

```
helm fetch https://helm.ngc.nvidia.com/nvaie/charts/network-operator-v1.1.0.tgz --
username='$oauthtoken' --password=<YOUR API KEY> --untar
```

4 Cree un archivo YAML para los valores de configuración.

```
vi values.yaml
```

5 Rellene el archivo `values.yaml` con la siguiente información.

```
deployCR: true
 ofedDriver:
  deploy: true
 rdmaSharedDevicePlugin:
  deploy: true
  resources:
   - name: rdma_shared_device_a
   vendors: [15b3]
   devices: [ens192]
```

6 Instale el operador de redes de NVIDIA mediante el siguiente comando.

```
helm install network-operator -f ./values.yaml -n network-operator --create-namespace --
wait network-operator/
```

## Paso 4 del operador: Instalar el operador de GPU de NVIDIA

NVIDIA proporciona un operador de GPU preconfigurado para los clientes de NVIDIA AI Enterprise. En estas instrucciones se supone que está utilizando esta versión preconfigurada del operador de GPU. Estas instrucciones se basan en las instrucciones que proporciona NVIDIA para instalar el operador de GPU, pero que se actualizaron para TKG en vSphere 8.

Complete los siguientes pasos para instalar el operador de GPU NVIDIA AI Enterprise en el clúster de TKG que aprovisionó.

1 Compruebe que haya iniciado sesión en el clúster de carga de trabajo de vGPU de TKGS y que el contexto esté establecido en el espacio de nombres del clúster de carga de trabajo correspondiente a la vGPU de TKGS.

   Consulte las instrucciones Paso 2 del operador: Aprovisionar un clúster de TKGS para vGPU de NVIDIA si es necesario.

2 Para instalar Helm, consulte la documentación de Helm, si aún no está instalado.

3 Cree el espacio de nombres de Kubernetes `gpu-operator`.

```
kubectl create namespace gpu-operator
```

4 Cree un archivo de configuración de licencia de vGPU vacío.

```
sudo touch gridd.conf
```

5 Genere y descargue un token de licencia de cliente NLS.

Consulte la *Sección 4.6. Generar un token de configuración de cliente* de la Guía de usuario del sistema de licencias de NVIDIA.

6 Cambie el nombre del token de licencia de cliente NLS que descargó en `client_configuration_token.tok`.

7 Cree el objeto `licensing-config` de ConfigMap en el espacio de nombres `gpu-operator`.

Incluya el archivo de configuración de licencia de vGPU (`gridd.conf`) y el token de licencia de cliente NLS (`*.tok`) en este ConfigMap

```
kubectl create configmap licensing-config \
    -n gpu-operator --from-file=gridd.conf --from-file=<path>/
client_configuration_token.tok
```

8 Cree un secreto de extracción de imágenes para el registro privado que contenga el controlador de gráficos de software vGPU NVIDIA en contenedor para Linux para usarlo con el operador de GPU NVIDIA.

Cree el secreto de extracción de imágenes en el espacio de nombres `gpu-operator` con el nombre del secreto de registro `ngc-secret` y el nombre del registro privado `nvcr.io/nvaie`. Incluya la clave de API de NGC y la dirección de correo electrónico en los campos que se indican.

```
kubectl create secret docker-registry ngc-secret \
--docker-server='nvcr.io/nvaie' \
--docker-username='$oauthtoken' \
--docker-password=<YOUR_NGC_API_KEY> \
--docker-email=<YOUR_EMAIL_ADDRESS> \
-n gpu-operator
```

9 Descargue el gráfico de Helm para la versión 2.2 del operador de GPU NVAIE.

Reemplace YOUR API KEY.

```
helm fetchhttps://helm.ngc.nvidia.com/nvaie/charts/gpu-operator-2-2-v1.11.1.tgz--
username='$oauthtoken' \
--password=<YOUR API KEY>
```

10 Instale la versión 2.2 del operador de GPU NVAIE en el clúster de TKG.

```
helm install gpu-operator ./gpu-operator-2-2-v1.11.1.tgz -n gpu-operator
```

## Paso 5 del operador: Implementar una carga de trabajo de AI/ML

El catálogo de NVIDIA GPU Cloud ofrece varias imágenes de contenedor que se encuentran disponibles para ejecutar cargas de trabajo de AI/ML en los clústeres de Tanzu Kubernetes habilitados para vGPU. Para obtener más información sobre las imágenes disponibles, consulte la documentación de NGC.

# Crear una clase de máquina virtual personalizada para dispositivos vGPU de NVIDIA

Consulte este tema para crear una clase de máquina virtual personalizada para dispositivos vGPU de NVIDIA Grid.

## Crear una clase de máquina virtual personalizada con perfil de vGPU (v8 U2 P03 y versiones posteriores)

La unidad de procesamiento de gráficos virtuales (Virtual Graphics Processing Unit, vGPU) de NVIDIA permite que varias máquinas virtuales compartan una sola GPU física. Para utilizar vGPU con clústeres de TKGS, defina una clase de máquina virtual personalizada. A partir de esta versión, hay un nuevo asistente para definir las clases de máquinas virtuales personalizadas. A diferencia del Crear una clase de máquina virtual personalizada con un perfil de vGPU (v8 U2 y versiones anteriores) para definir una clase de máquina virtual personalizada, el perfil de vGPU se lee desde el dispositivo en lugar de configurarse en la clase de máquina virtual.

El operador de máquina virtual sondea el inventario de vCenter para obtener todos los dispositivos vGPU instalados en hosts ESXi que componen el clúster de vSphere en el que se implementa Supervisor. El dispositivo vGPU define su perfil. El nombre del dispositivo vGPU indica si el perfil es una GPU de varias instancias (MIG) o una GPU de tiempo compartido. La MIG corta los recursos informáticos y permite que varias cargas de trabajo se ejecuten en paralelo en una sola GPU. El uso compartido de tiempo proporciona acceso compartido a la GPU. El modo MIG se basa en una arquitectura de GPU más reciente y solo se admite en dispositivos NVIDIA A100 y A30. Consulte la documentación de NVIDIA para obtener más información.

Por ejemplo, el dispositivo GPU "grid-a100-40c" proporciona un perfil de vGPU de tiempo compartido que asigna un dispositivo GPU NVIDIA A100 con 40 GB de memoria a una máquina virtual. El perfil de vGPU basado en MIG equivalente sería el dispositivo "grid-a100-7-40c". Puede identificar que se trata de un perfil de MIG porque hay un número adicional entre el dispositivo y la memoria RAM. El valor "7" indica que hay 7 sectores informáticos en el dispositivo GPU. Los perfiles de vGPU basados en MIG pueden tener 1, 2, 3 o 7 sectores informáticos.

1 En el menú Inicio de vSphere Client, seleccione **Administración de cargas de trabajo > Servicios**.

2 Seleccione la pestaña **Clases de máquina virtual**.

3 Haga clic en **Crear clase de VM**.

Esta acción inicia el asistente Crear clase de máquina virtual, que servirá de guía para la creación de una clase de máquina virtual.

4 En **Nombre**, introduzca un nombre para la clase de máquina virtual y haga clic en **Siguiente**.

El nombre de la clase de máquina virtual identifica la clase de máquina virtual. Introduzca un nombre único conforme con DNS que cumpla estos requisitos:

- Utilice un nombre único que no sea un duplicado de los nombres de las clases de máquinas virtuales predeterminadas o personalizadas de su entorno.
- Utilice una cadena alfanumérica con una longitud máxima de 63 caracteres.
- No utilice caracteres en mayúscula ni espacios.
- Puede utilizar guiones en cualquier lugar, excepto como primer o último carácter. Por ejemplo, **`vm-class1`**.
- Después de crear la clase de máquina virtual, no puede cambiarle el nombre.

5 Para **Compatibilidad**, seleccione **ESXi 8.0 U2 y versiones posteriores** y haga clic en **Siguiente**.

Para obtener más información, consulte Compatibilidad de máquinas virtuales.

**Nota** No se puede cambiar la compatibilidad de hardware de una clase de máquina virtual después de crearla.

6 En **Configuración > Hardware virtual**, agregue el dispositivo GPU de NVIDIA a la clase de máquina virtual.

a Seleccione **Configuración > Hardware virtual > Agregar nuevo dispositivo > Dispositivo PCI**.

b Seleccione el dispositivo vGPU de NVIDIA Grid deseado de la lista. Existen dos tipos de perfiles vGPU de NVIDIA Grid: **Uso compartido de tiempo** y **Uso compartido de GPU de varias instancias**. El sistema detecta el perfil cuando se selecciona el dispositivo.

**Nota** Solo puede agregar un dispositivo vGPU de NVIDIA GRID del tipo perfil de MIG a una clase de máquina virtual.

c Haga clic en **Seleccionar**, y **Nuevo dispositivo PCI** aparecerá en la pestaña Hardware virtual.

7 En **Configuración > Hardware virtual**, especifique la configuración deseada para **CPU**, **Memoria**, **Nuevo dispositivo PCI**, **Tarjeta de video** y **Dispositivos de seguridad**.

Tabla 13-3. Configuración de CPU

| Configuración | Configuración |
| --- | --- |
| CPU | Seleccione el número de CPU virtuales para la máquina virtual. Consulte Configuración y limitaciones de la CPU virtual para obtener más información. |
| Topología de CPU | Asignado al encender |
| Reserva | La reserva debe tener entre 0 y 10 MHz |
| Límite | El límite debe ser mayor o igual que 10 MHz |
| Recursos compartidos | Las opciones son Bajo, Normal, Alto y Personalizado |
| Virtualización de hardware | Seleccione esta opción para exponer la virtualización asistida por hardware en el SO invitado |
| Contadores de rendimiento | Habilitar contadores de rendimiento virtualizados de la CPU |
| Afinidad de programación | Seleccione la afinidad de procesador físico para esta máquina virtual. Utilice "-" para indicar rangos y "," para separar valores. Por ejemplo, "0, 2, 4-7" indicaría los procesadores 0, 2, 4, 5, 6 y 7. Borre la cadena para quitar la configuración de afinidad. |
| I/O MMU | Seleccione esta opción para habilitar la unidad de administración de memoria (página a disco) |

Tabla 13-4. Configuración de memoria

| Configuración | Configuración |
| --- | --- |
| Memoria | Seleccione el tamaño de la memoria para la máquina virtual. Consulte Memoria máxima de la máquina virtual para obtener más información. |
| Reserva | Especifique la asignación mínima garantizada para una máquina virtual o reserve toda la memoria de invitado. Si no se puede cumplir la reserva, la máquina virtual no se puede ejecutar. |
| Límite | Seleccione la cantidad de memoria que se debe limitar para establecer un límite en el uso de memoria de una máquina virtual. |
| Recursos compartidos | Seleccione la cantidad de memoria que desea compartir. Las cuotas representan una métrica relativa para la asignación de capacidad de memoria. Para obtener más información, consulte Uso compartido de la memoria. |
| Conexión en caliente de memoria | Habilite (marque la casilla) para permitir la adición de recursos de memoria a una máquina virtual que esté encendida. ConsulteConfiguración de adición de memoria en caliente para obtener más detalles. |

### Tabla 13-5. Nueva configuración de uso compartido de dispositivo PCI > GPU

| Modo de Uso compartido de tiempo | Modo MIG |
| --- | --- |
| En el modo de Uso compartido de tiempo, el programador de vGPU indica a la GPU que realice el trabajo para cada máquina virtual habilitada para vGPU en serie durante un período de tiempo con el mejor objetivo de esfuerzo de equilibrar el rendimiento entre las vGPU. | El modo MIG permite que varias máquinas virtuales habilitadas para vGPU se ejecuten en paralelo en un solo dispositivo GPU. Si no ve la opción MIG, el dispositivo PCI que seleccionó no lo admite. |

### Tabla 13-6. Configurar tarjeta de video

| Configuración | Configuración |
| --- | --- |
| Tarjeta de vídeo | Elija si desea detectar automáticamente los ajustes desde el hardware o introducir ajustes personalizados. Si selecciona la detección automática, no se pueden configurar otros ajustes. |
| Número de pantallas | Seleccione el número de pantallas. |
| Memoria total de vídeo | Introduzca la memoria total de video, en MB. |
| Gráficos 3D | Seleccione esta opción para habilitar la compatibilidad con 3D. |

### Tabla 13-7. Configurar dispositivos de seguridad

| Configuración | Configuración |
| --- | --- |
| Dispositivo de seguridad | Si el dispositivo de seguridad de SGX está instalado, puede configurar los ajustes de la máquina virtual aquí; de lo contrario, este campo no se puede configurar. Consulte la documentación de SGX para obtener más detalles. |

8 Seleccione la pestaña **Configuración > Opciones de máquina virtual** y configure los ajustes adicionales de la máquina virtual. Consulte Configurar opciones de máquina virtual para obtener instrucciones.

9 Seleccione la pestaña **Configuración > Parámetros avanzados** y agregue atributos para la clase de máquina virtual.

10 Haga clic en **Siguiente**.

11 En la página **Revisar y confirmar**, revise los detalles y haga clic en **Finalizar**.

12 Asocie la nueva clase de máquina virtual con el espacio de nombres de vSphere. Consulte Asociar las clases de máquinas virtuales con el espacio de nombres de vSphere.

Figura 13-1. Selección de dispositivo vGPU de NVIDIA

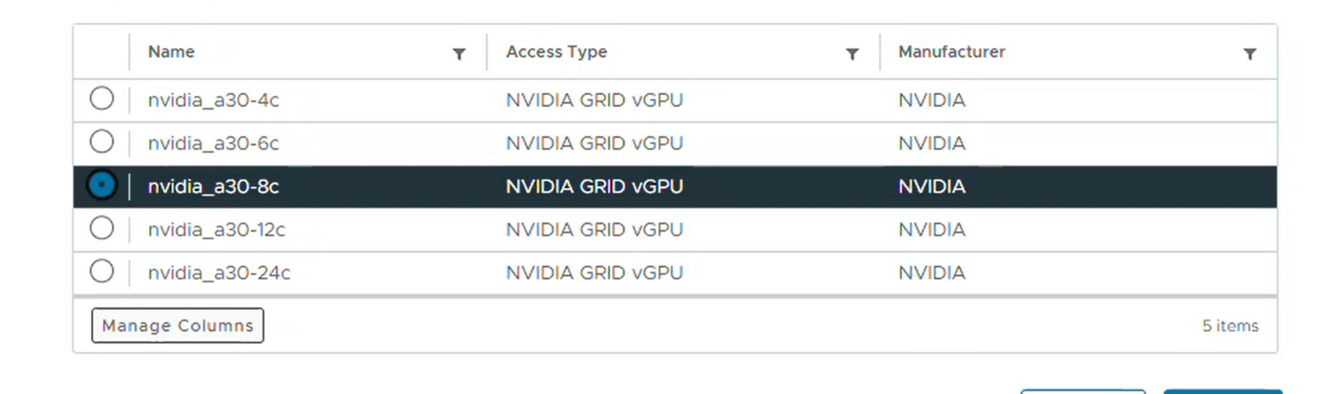
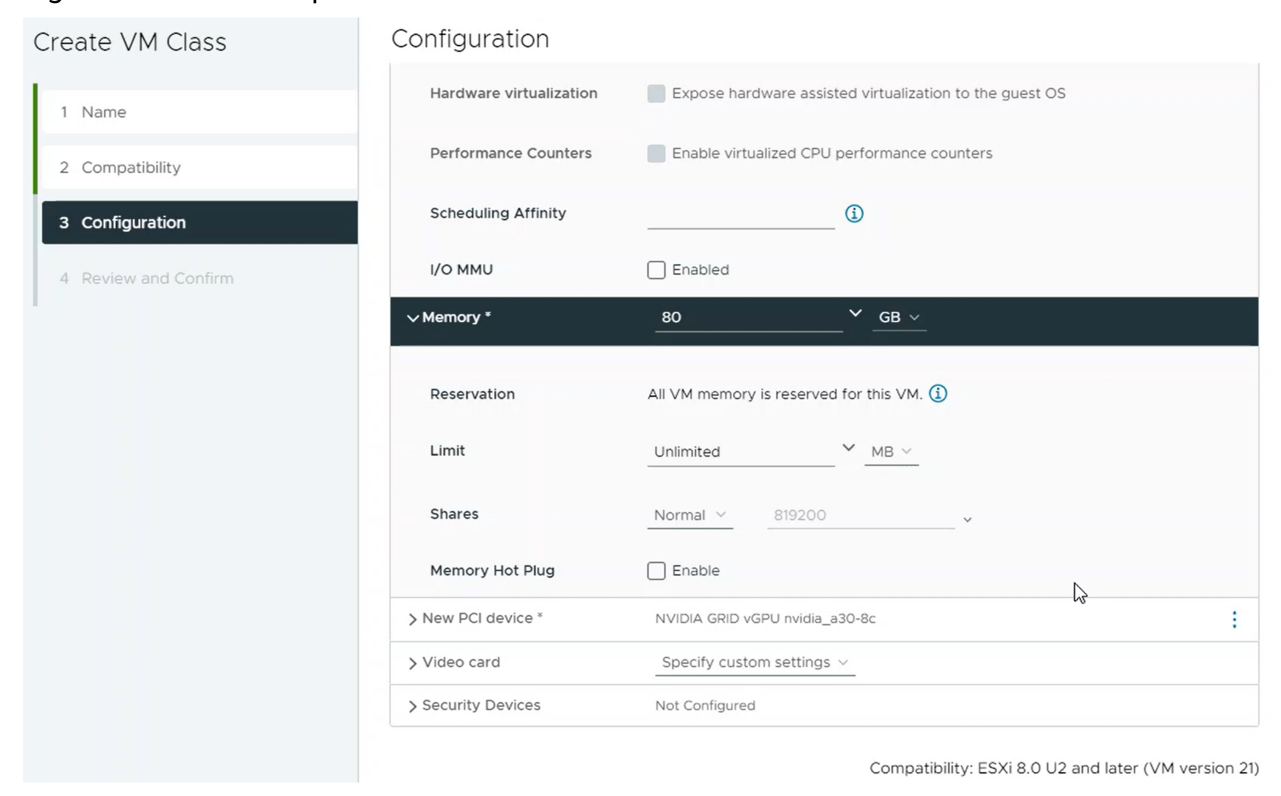



Figura 13-2. Nuevo dispositivo PCI de vGPU de NVIDIA




## Crear una clase de máquina virtual personalizada con un perfil de vGPU (v8 U2 y versiones anteriores)

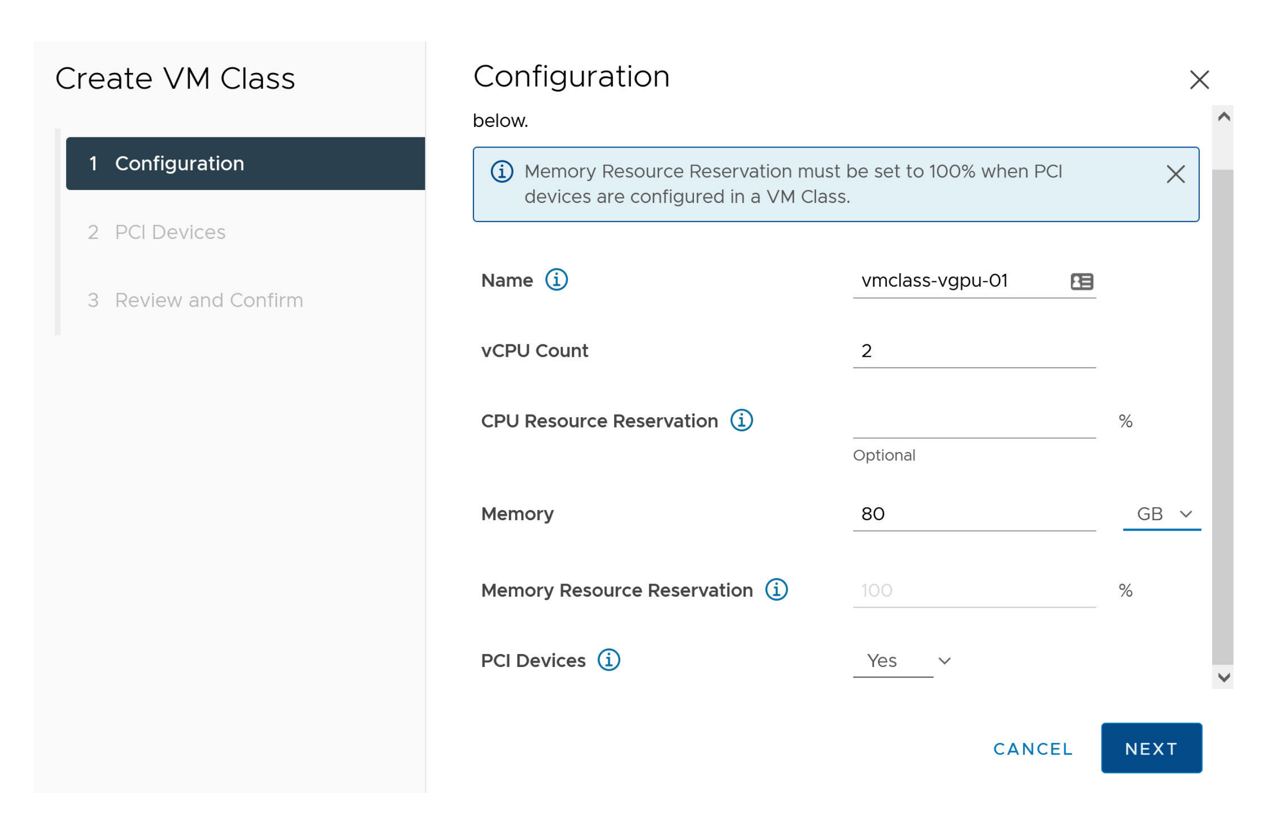
El siguiente paso consiste en crear una clase de máquina virtual personalizada con un perfil de vGPU. El sistema utilizará esta definición de clase cuando cree los nodos del clúster de TKG.

Siga las instrucciones siguientes para crear una clase de máquina virtual personalizada con un perfil de vGPU.

1 Inicie sesión en vCenter Server con vSphere Client.
2 Seleccione **Administración de cargas de trabajo**.
3 Seleccione **Servicios**
4 Seleccione **Clases de VM**.
5 Haga clic en **Crear clase de VM**.
6 En la pestaña **Configuración**, configure la clase de máquina virtual personalizada.

| Campo de configuración | Descripción |
| --- | --- |
| **Nombre** | Introduzca un nombre descriptivo para la clase de máquina virtual personalizada, como `vmclass-vgpu-1`. |
| **Recuento de vCPU** | **2** |
| **Reserva de recursos de CPU** | Opcional, acepte para dejar en blanco |
| **Memoria** | **80 GB**, por ejemplo |
| **Reserva de recursos de memoria** | **100 %** (obligatorio cuando se configuran dispositivos PCI en una clase de máquina virtual) |
| **Dispositivos PCI** | **Sí**<br>**Nota** Al seleccionar Sí para Dispositivos PCI, se indica al sistema que se utiliza un dispositivo GPU y se cambia la configuración de la clase de máquina virtual para admitir la configuración de vGPU. Para obtener más información, consulte Agregar dispositivos PCI a una clase de máquina virtual en vSphere with Tanzu. |

Por ejemplo:



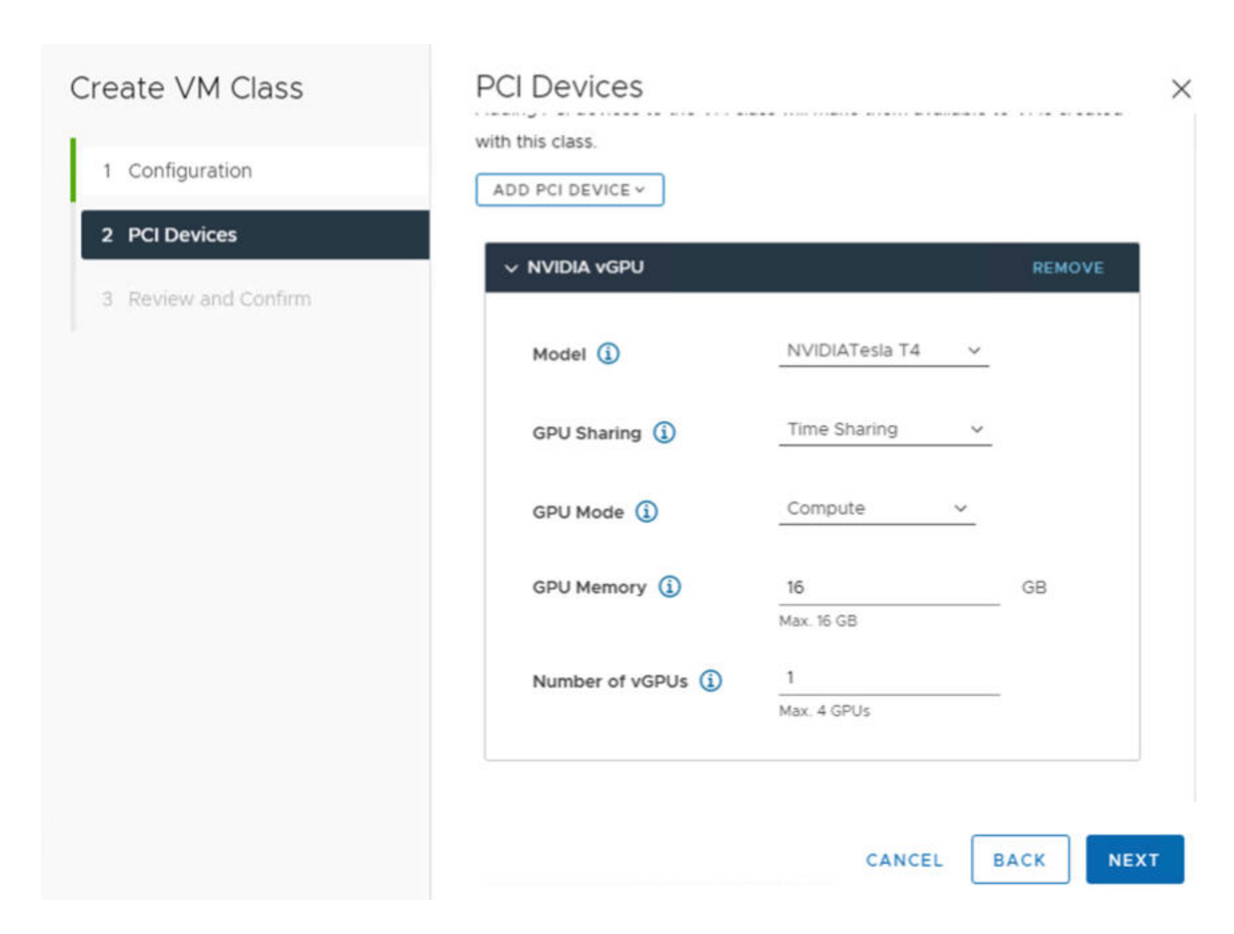

7 Haga clic en **Siguiente**.

8 En la pestaña **Dispositivos PCI**, seleccione la opción **Agregar dispositivo PCI > vGPU de NVIDIA**.

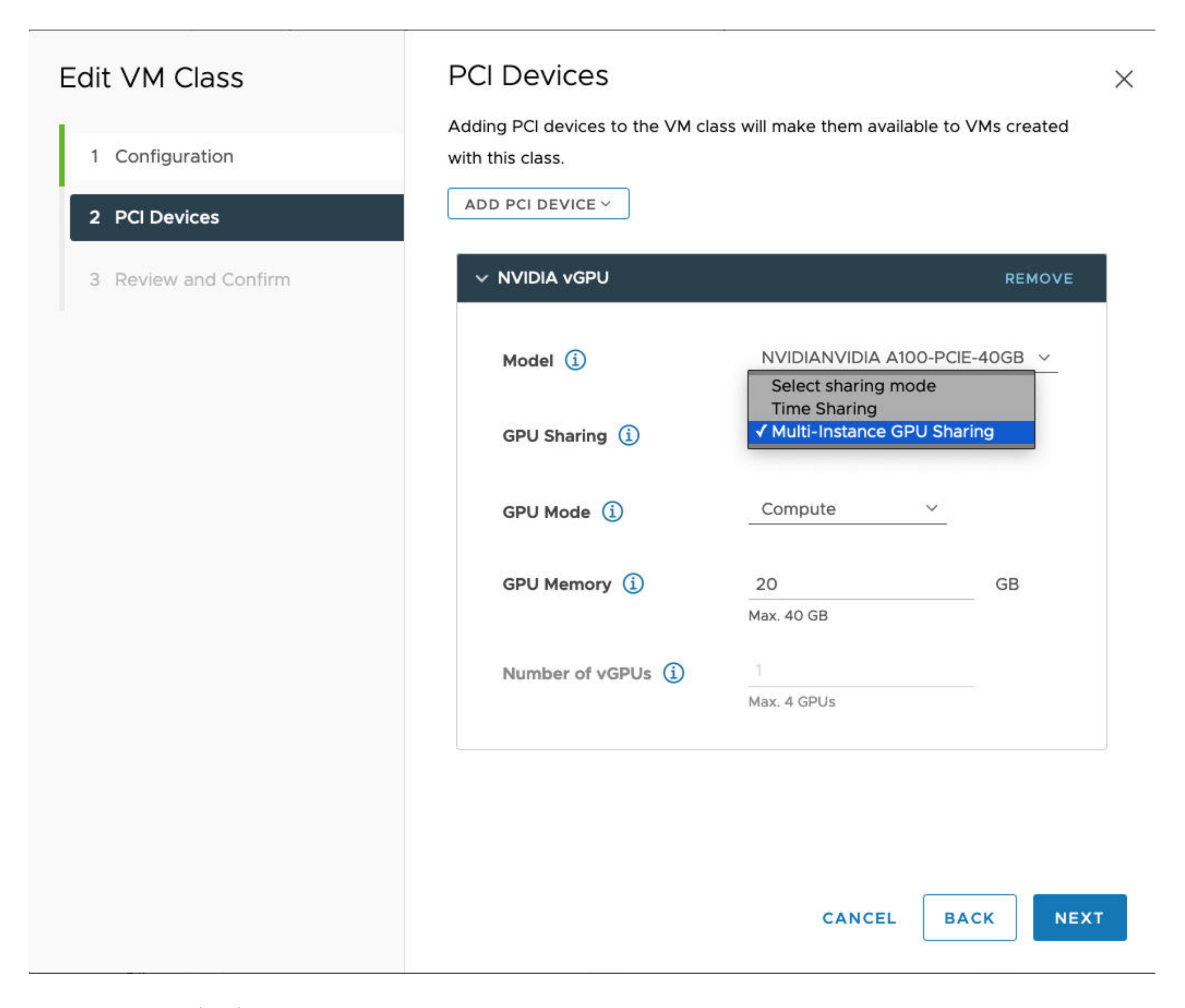
9 Configure el modelo NVIDIA vGPU.

| Campo NVIDIA vGPU | Descripción |
| --- | --- |
| **Modelo** | Seleccione el modelo del dispositivo de hardware GPU NVIDIA de los disponibles en el menú **vGPU de NVIDIA > Modelo**. Si el sistema no muestra ningún perfil, ninguno de los hosts del clúster tiene dispositivos PCI compatibles. |
| **Uso compartido de GPU** | Este ajuste define cómo se comparte el dispositivo GPU entre máquinas virtuales habilitadas para GPU. Existen dos tipos de implementaciones de vGPU: **Uso compartido de tiempo** y **Uso compartido de GPU de varias instancias**.<br>En el modo de Uso compartido de tiempo, el programador de vGPU indica a la GPU que realice el trabajo para cada máquina virtual habilitada para vGPU en serie durante un período de tiempo con el mejor objetivo de esfuerzo de equilibrar el rendimiento entre las vGPU.<br>El modo MIG permite que varias máquinas virtuales habilitadas para vGPU se ejecuten en paralelo en un solo dispositivo GPU. El modo MIG se basa en una arquitectura de GPU más reciente y solo se admite en dispositivos NVIDIA A100 y A30. Si no ve la opción MIG, el dispositivo PCI que seleccionó no lo admite. |
| **Modo GPU** | **Cálculo** |
| **Memoria de GPU** | **8 GB**, por ejemplo |
| **Número de vGPU** | **1**, por ejemplo |

Por ejemplo, este es un perfil de NVIDIA vGPU configurado en el modo Uso compartido de tiempo:




Por ejemplo, aquí se muestra un perfil de NVIDIA vGPU configurado en el modo MIG con un dispositivo GPU compatible:




10 Haga clic en **Siguiente**.

11 Revise y confirme las selecciones que hizo.

12 Haga clic en **Finalizar**.

13 Compruebe que la nueva clase de máquina virtual personalizada esté disponible en la lista de clases de máquinas virtuales.

### vGPU con Instancia dinámica de DirectPath I/O

Si utiliza vGPU con Instancia dinámica de DirectPath I/O, complete la siguiente configuración adicional. Agregue una segunda configuración de dispositivo PCI a la clase de máquina virtual personalizada que creó con **Instancia dinámica de DirectPath I/O** especificada y el dispositivo PCI compatible seleccionado. Cuando se crea una instancia de una clase de máquina virtual de este tipo, vSphere Distributed Resource Scheduler (DRS) determina la colocación de la máquina virtual.

1 Seleccione **Administración de cargas de trabajo**.

2 Seleccione **Servicios**.

3 Seleccione **Clases de VM**.

4 Edite la clase de máquina virtual personalizada que ya está configurada con el perfil de **NVIDIA vGPU**.

5 Seleccione la pestaña **Dispositivos PCI**.

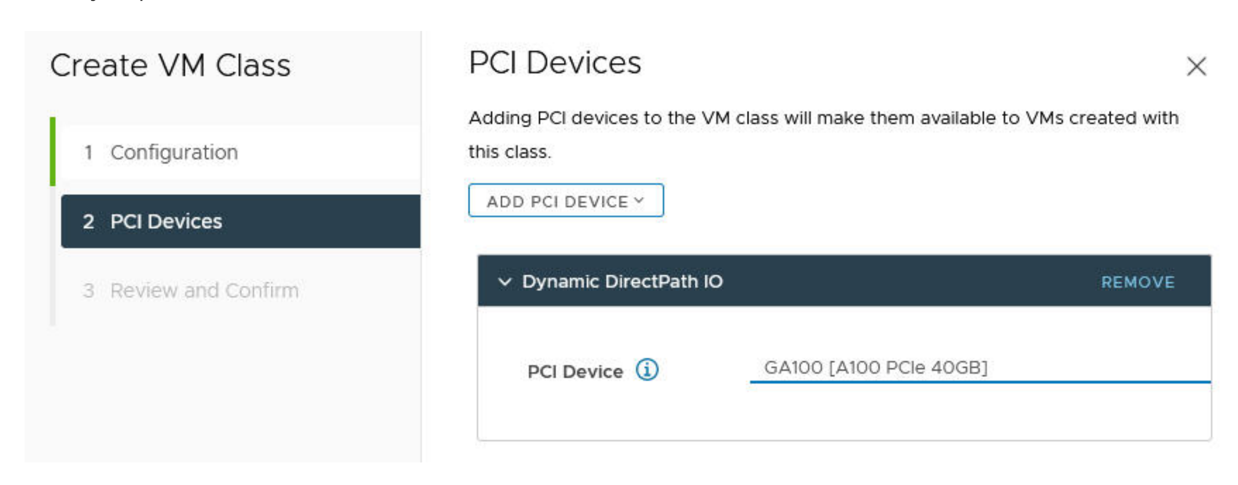
6 Haga clic en **Agregar dispositivo PCI**.

7 Seleccione la opción **Instancia dinámica de DirectPath I/O**.




8 Seleccione el **Dispositivo PCI**.

Por ejemplo:




9 Haga clic en **Siguiente**.

10 Revise y confirme las selecciones que hizo.

11 Haga clic en **Finalizar**.

12 Compruebe que la nueva clase de máquina virtual personalizada esté disponible en la lista de clases de máquinas virtuales.

# Uso de registros privados con clústeres Servicio TKG

14

Los registros de contenedor proporcionan a los operadores de Kubernetes un repositorio adecuado para almacenar y compartir imágenes de contenedor. En esta sección se proporciona documentación sobre el uso de registros privados con clústeres de Servicio TKG.

Lea los siguientes temas a continuación:

- Integrar clústeres de Servicio TKG con un registro de contenedor privado
- Crear secreto de credencial de registro privado
- Crear un pod a partir de una imagen de contenedor en un registro privado
- Instalar el servicio Supervisor de Harbor
- Configurar un cliente de Docker con el certificado de registro de Harbor
- Insertar paquetes estándar en un registro de Harbor privado

## Integrar clústeres de Servicio TKG con un registro de contenedor privado

Consulte este tema para integrar clústeres de Servicio TKG con un registro de contenedor privado.

### Caso de uso del registro de contenedor privado

Los registros de contenedores proporcionan una función crítica para las implementaciones de Kubernetes, y sirven como repositorio centralizado para almacenar y compartir imágenes de contenedor. El registro de contenedor público utilizado con más frecuencia es Docker Hub. Existen muchas ofertas de registros de contenedores privados. VMware Harbor es un registro de contenedor privado, nativo en la nube y de código abierto que viene con Supervisor.

## Integración del registro de contenedor privado

Para integrar un registro privado con un clúster de Servicio TKG, configure el clúster con uno o más certificados de CA autofirmados para que proporcione contenido de registro privado a través de HTTPS. Para ello, en la especificación del clúster se incluye un campo `trust` con el valor `additionalTrustedCAs`. Puede definir cualquier número de certificados de CA autofirmados en los que el clúster de TKGS debe confiar. Esta funcionalidad permite definir fácilmente una lista de certificados y actualizar los que necesiten rotación.

Puede configurar el certificado de registro privado la primera vez que crea el clúster o puede actualizar un clúster existente y proporcionar el certificado del registro privado. Para editar un clúster existente y agregar el certificado del registro privado, utilice el método de `kubectl edit`. Consulte Configurar un editor de texto para Kubectl.

Tenga en cuenta que la implementación del campo `trust.additionalTrustedCAs` difiere ligeramente entre las API compatibles para aprovisionar clústeres de TKGS. Consulte las tablas Tabla 14-1. Campos de confianza de API v1alpha3 y Tabla 14-2. Variable de confianza de la API v1beta1 para obtener más información.

## Ejemplo de API v1alpha3

El siguiente ejemplo demuestra cómo integrar un clúster de Servicio TKG aprovisionado mediante la API v1alpha3 con un registro de contenedor privado usando su certificado de CA.

Con el API v1alpha3 del clúster de Tanzu Kubernetes, el campo `trust.additionalTrustedCAs` incluye uno o varios pares nombre-datos, cada uno de los cuales puede incluir un certificado de TLS para un registro privado.

### Tabla 14-1. Campos de confianza de API v1alpha3

| Campo | Descripción |
|---|---|
| `trust` | Marcador de sección. No acepta datos. |
| `additionalTrustedCAs` | Marcador de sección. Incluye una matriz de certificados con campos de `name` y `data` para cada uno. |
| `name` | Nombre definido por el usuario del certificado de CA. |
| `data` | Contenido del certificado de CA (`ca.crt`) en formato PEM que está doblemente codificado en base64.<br>**Nota** La API v1alpha3 requiere que el contenido del certificado esté codificado en base64 único. Si el contenido no tiene codificación base6 única, no se puede procesar el archivo PEM resultante. |

Utilice el siguiente procedimiento para integrar Harbor con un clúster de API v1alpha3 mediante el certificado del registro de Harbor.

1 Descargue el **certificado de registro** de Harbor desde la interfaz web de Harbor en la pantalla **Proyectos > Repositorios**.

El archivo de certificado de CA se descarga como `ca.crt`.

2 El contenido del certificado de CA se codifica con una sola base64.

- Linux: `base64 -w 0 ca.crt`
- Windows: https://www.base64encode.org/.

3 Incluya los campos `trust.additionalTrustedCAs` en la especificación del clúster y rellene con los valores `name` y `data`.

```
#tkc-with-trusted-private-reg-cert.yaml
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TanzuKubernetesCluster
metadata:
  name: tkc01
  namespace: tkgs-cluster-ns
spec:
   topology:
     controlPlane:
       replicas: 3
       storageClass: tkgs-storage-policy
       vmClass: guaranteed-medium
       tkr:
         reference:
           name: v1.25.7---vmware.3-fips.1-tkg.1
     nodePools:
     - name: nodepool-01
       replicas: 3
       storageClass: tkgs-storage-policy
       vmClass: guaranteed-medium
       tkr:
         reference:
           name: v1.25.7---vmware.3-fips.1-tkg.1
   settings:
     storage:
       defaultClass: tkgs-storage-policy
     network:
       cni:
         name: antrea
       services:
         cidrBlocks: ["198.51.100.0/12"]
       pods:
         cidrBlocks: ["192.0.2.0/16"]
       serviceDomain: cluster.local
       trust:
         additionalTrustedCAs:
         - name: CompanyInternalCA-1
           data: LS0tLS1C...LS0tCg==
         - name: CompanyInternalCA-2
           data: MTLtMT1C...MT0tPg==
```

4 Para rotar un certificado, `kubectl edit` la especificación del clúster y actualice el valor de `trust.additionalTrustedCAs.data` y, a continuación, inicie una actualización gradual.

## Ejemplo de API v1beta1

En el siguiente ejemplo, se describe cómo integrar un clúster de Servicio TKG aprovisionado mediante la API v1beta1 con un registro de contenedor privado usando su certificado de CA.

Para integrar un registro de contenedor privado con un clúster de TKGS aprovisionado con el API de clúster v1beta1 , utilice la variable trust y rellénelo con un secreto de Kubernetes que contenga el certificado del registro privado.

**Tabla 14-2. Variable de confianza de la API v1beta1**

| Campo | Descripción |
| --- | --- |
| `trust` | Marcador de sección. No acepta datos. |
| `additionalTrustedCAs` | Marcador de sección. Incluye una matriz de certificados con el nombre para cada uno. |
| `name` | El nombre definido por el usuario para el campo de asignación de `data` en el secreto de Kubernetes que contiene el certificado de CA en formato PEM con codificación base64 doble.<br>**Nota** La API v1beta1 requiere que el contenido del certificado tenga doble codificación base64. Si el contenido no tiene doble codificación base6, no se puede procesar el archivo PEM resultante. |

Utilice el siguiente procedimiento para integrar Harbor con un clúster de API v1beta1 mediante el certificado del registro de Harbor.

1 Descargue el **certificado de registro** de Harbor desde la interfaz web de Harbor en la pantalla **Proyectos > Repositorios**.

   El archivo de certificado de CA se descarga como `ca.crt`.

2 Codificación base64 doble del contenido del certificado de CA.

   - Linux: `base64 -w 0 ca.crt | base64 -w 0`
   - Windows: https://www.base64encode.org/.

3 Cree un archivo YAML de definición de secreto de Kubernetes con el siguiente contenido.

```
#additional-ca-1.yaml
apiVersion: v1
data:
  additional-ca-1: TFMwdExTMUNSGlSzZ3Jaa...VVNVWkpRMEMwdExTMHRDZz09
kind: Secret
metadata:
  name: cluster01-user-trusted-ca-secret
  namespace: tkgs-cluster-ns
type: Opaque
```

Donde:

- El valor de la asignación de `data` del secreto es una cadena definida por el usuario que es el nombre del certificado de CA (`additional-ca-1` en este ejemplo) cuyo valor es un certificado con codificación base64 doble.
- En la sección `metadata`, asigne el nombre *`CLUSTER-NAME`*`-user-trusted-ca-secret` al secreto, donde *CLUSTER-NAME* es el nombre del clúster. Este secreto se debe crear en el mismo espacio de nombres de vSphere que el clúster.

4 Cree el secreto de Kubernetes de forma declarativa.

```
kubectl apply -f additional-ca-1.yaml
```

5 Compruebe la creación del secreto.

```
kubeclt get secret -n tkgs-cluster-ns cluster01-user-trusted-ca-secret
NAME                                             TYPE     DATA   AGE
cluster01-user-trusted-ca-secret                 Opaque   12     2d22h
```

6 Incluya la variable `trust` en la especificación del clúster que hace referencia el nombre de la asignación de datos del secreto, que en este ejemplo es `additional-ca-1`.

```
#cluster-with-trusted-private-reg-cert.yaml
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: cluster01
  namespace: tkgs-cluster-ns
spec:
  clusterNetwork:
    services:
      cidrBlocks: ["198.52.100.0/12"]
    pods:
      cidrBlocks: ["192.101.2.0/16"]
    serviceDomain: "cluster.local"
  topology:
    class: tanzukubernetescluster
    version: v1.26.5+vmware.2-fips.1-tkg.1
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: node-pool
          name: node-pool-01
          replicas: 3
    variables:
      - name: vmClass
        value: guaranteed-medium
      - name: storageClass
        value: tkgs-storage-profile
      - name: defaultStorageClass
        value: tkgs-storage-profile
```

```
    - name: trust
      value:
        additionalTrustedCAs:
        - name: additional-ca-1
```

7 Para rotar el certificado, cree un nuevo secreto y edite la especificación del clúster con el valor adecuado. Esto activará una actualización gradual del clúster.

**Nota** El sistema no supervisa los cambios en *CLUSTER-NAME*-user-trusted-ca-secret. Si cambia el valor de asignación de data, estos cambios no se verán reflejados en el clúster. Deberá crear un nuevo secreto y su asignación de datos name en trust.additionalTrustCAs.

# Crear secreto de credencial de registro privado

Cree un secreto de credencial de registro y una referencia en las especificaciones de pods y de implementación para poder extraer imágenes de contenedor sin errores.

Docker Hub requiere una cuenta desde la que extraer imágenes. Puede crear un secreto de Kubernetes con sus credenciales de Docker Hub y hacer referencia a este secreto en los pods y las especificaciones de implementación.

Este enfoque también se puede utilizar para otros registros privados.

### Requisitos previos

Instale Docker en una máquina cliente Ubuntu. Consulte https://docs.docker.com/engine/install/ubuntu/.

### Procedimiento

1 Ejecute `docker version` y compruebe que Docker esté instalado.

2 Ejecute `docker login -u USERNAME`.

3 Introduzca la contraseña de Docker Hub.

4 Ejecute `cat ~/.docker/config.json`.

5 Ejecute el siguiente comando para crear un secreto genérico denominado `regcred` que incluya las credenciales de acceso de Docker Hub.

```
kubectl create secret generic regcred \
    --from-file=.dockerconfigjson=/home/ubuntu/.docker/config.json \
    --type=kubernetes.io/dockerconfigjson
```

Debería ver `secret/regcred created`.

6 Compruebe el secreto.

```
kubectl get secrets

NAME                  TYPE                                  DATA   AGE
default-token-w7wqk   kubernetes.io/service-account-token   3      6h28m
regcred               kubernetes.io/dockerconfigjson        1      3h22m
```

Haga referencia al secreto regcred en el YAML.

7 Haga referencia al secreto `regcred` en la sección `imagePullSecrets` de la especificación de implementación o de pod para las imágenes de contenedor.

Por ejemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: ping-pod
  namespace: default
spec:
  containers:
  - image: busybox:1.34
    name: busybox
    command: ["ping", "-c"]
    args: ["1", "8.8.8.8"]
  imagePullSecrets:
  - name: regcred
  restartPolicy: Never
```

# Crear un pod a partir de una imagen de contenedor en un registro privado

Si va a extraer imágenes de un registro de contenedor privado para un clúster de Servicio TKG, configure el YAML de carga de trabajo con los detalles del registro privado.

Este procedimiento se puede utilizar para extraer imágenes de un registro de contenedor privado, como registro de Harbor. En este ejemplo, creamos una especificación de pod que utilizará una imagen almacenada en un registro de Harbor y utilizará el secreto de extracción de imágenes que se configuró anteriormente.

### Requisitos previos

Cree un secreto de credencial de registro. Consulte Crear secreto de credencial de registro privado.

### Procedimiento

1 Aprovisione un clúster de TKG.

Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

2 Inicie sesión en el clúster.

Consulte Conectarse a un clúster de Servicio TKG como usuario de vCenter Single Sign-On con Kubectl.

3 Cree un secreto de credencial de registro.

Consulte Crear secreto de credencial de registro privado.

4 Cree un ejemplo de especificación de Pod con los detalles del registro privado.

`pod-example.yaml`

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
  - name: private-reg-container
    image: <Registry-IP-or-FQDN>/<vsphere-namespace>/<image-name>:<version>
  imagePullSecrets:
  - name: <registry-secret-name>
```

- Reemplace `<workload-name>` por el nombre de la carga de trabajo del pod.
- Reemplace `<kubernetes-namespace>` por el espacio de nombres de Kubernetes del clúster en el que se creará el pod. Debe ser el mismo espacio de nombres de Kubernetes en el que se almacena el secreto de extracción de imágenes del servicio de registro en el clúster de Tanzu Kubernetes (por ejemplo, el espacio de nombres predeterminado).
- Reemplace `<Registry-IP-or-FQDN>` por la dirección IP o el FQDN de la instancia de registro de Harbor integrada que se ejecuta en Supervisor.
- Reemplace *<vsphere-namespace>* por el espacio de nombres de vSphere en el que se aprovisiona el clúster de TKG de destino.
- Reemplace `<image-name>` por el nombre de imagen que desee.
- Reemplace `<version>` por una versión adecuada de la imagen (por ejemplo, "más reciente").
- Reemplace `<registry-secret-name>` por el nombre del secreto de extracción de imágenes del servicio de registro que creó anteriormente.

5 Cree la carga de trabajo con la especificación del pod que definió.

```
kubectl apply -f pod-example.yaml
```

El pod se debe crear a partir de la imagen extraída del registro.

# Instalar el servicio Supervisor de Harbor

Puede instalar el Registro de contenedores de Harbor como un servicio de Supervisor y ejecutar Harbor como un registro privado.

## Requisitos previos

Cumpla los siguientes requisitos previos:

- Supervisor de vSphere 8 está habilitado
- Conocer los servicios de Supervisor

**Nota** Estas instrucciones se validan con las redes vSphere 8 y NSX 4.

## Descargar archivos YAML requeridos

Descargue los archivos YAML necesarios, incluidos Contour y Harbor.

1 Descargue los siguientes archivos de Contour del sitio del servicio de controlador de entrada de Kubernetes:

   a El archivo de definición del servicio de Contour: `contour.yml`

   b El archivo de configuración del servicio de Contour: `contour-data-values.yml`

2 Descargue los siguientes archivos de Harbor del sitio del servicio de registro nativo en la nube:

   a El archivo de definición del servicio de Harbor: `harbor.yml`

   b El archivo de configuración del servicio de Harbor: `harbor-data-values.yml`

## Instalar Contour

Debe instalar Contour antes de instalar Harbor.

1 Cargue `contour.yml` en vCenter en **Administración de cargas de trabajo > > Servicios > Agregar**.

2 Compruebe que se ha añadido la definición del servicio de Contour.

3 Seleccione **Administración de cargas de trabajo > Supervisores > Supervisor > Configurar**.

4 Seleccione **Servicios de supervisor > Descripción general**.

5 Seleccione la pestaña **Disponible**.

6 Seleccione **Contour** y haga clic en **Instalar**.

7 Copie y pegue el contenido del archivo `contour-data-values.yml` en el campo de entrada "Configuración del servicio YAML".

   **Nota** Los valores de datos de Contour se pueden utilizar tal como están y no requieren cambios de configuración. El tipo de servicio para Envoy se establece en LoadBalancer.

8 Haga clic en **Aceptar** para continuar con la instalación de Contour.

9 Compruebe que Contour está instalado.

   a Seleccione el espacio de nombres de vSphere denominado *svc-contour-domain-XXXX*.

   b Seleccione la pestaña **Red** y después **Servicios**.

   c Debería ver el servicio Contour de tipo ClusterIP y el servicio Envoy de tipo LoadBalancer. El servicio Envoy debe tener una dirección IP externa.

## Actualizar valores de datos de Harbor

Antes de instalar Harbor, actualice el archivo de valores de datos.

1 En un editor de texto, abra el archivo `harbor-data-values.yml`.

2 Realice las siguientes modificaciones (como mínimo, editar otros campos es opcional).

3 Guarde los cambios.

| Nombre | Valor |
| --- | --- |
| *hostname* | harbordomain.com (elija algo único) |
| *tlsCertificate.tlsSecretLabels* | {"managed-by": "vmware-vRegistry"} (compruebe este valor, pero manténgalo tal como está) |
| *persistence.persistentVolumeClaim.registry.storageClass* | "vwt-storage-policy" (introduzca el nombre de la directiva de almacenamiento de vSphere para Supervisor) |
| *persistence.persistentVolumeClaim.jobservice.storageClass* | "vwt-storage-policy" (introduzca el nombre de la directiva de almacenamiento de vSphere para Supervisor) |
| *persistence.persistentVolumeClaim.database.storageClass* | "vwt-storage-policy" (introduzca el nombre de la directiva de almacenamiento de vSphere para Supervisor) |
| *persistence.persistentVolumeClaim.redis.storageClass* | "vwt-storage-policy" (introduzca el nombre de la directiva de almacenamiento de vSphere para Supervisor) |
| *persistence.persistentVolumeClaim.trivy.storageClass* | "vwt-storage-policy" (introduzca el nombre de la directiva de almacenamiento de vSphere para Supervisor) |

## Instalar Harbor

Complete las siguientes instrucciones para instalar Harbor.

1 Cargue `harbor.yml` en vCenter en **Administración de cargas de trabajo > > Servicios > Agregar**.

2 Compruebe que se ha añadido la definición del servicio de Harbor.

3 Seleccione **Administración de cargas de trabajo > Supervisores > Supervisor > Configurar**.

4 Seleccione **Servicios de supervisor > Descripción general**.

5 Seleccione la pestaña **Disponible**.

6 Seleccione **Harbor** y haga clic en **Instalar**.

7 Copie y pegue el contenido del archivo `harbor-data-values.yml` que editó en el campo de entrada "Configuración del servicio YAML".

8 Haga clic en **Aceptar** para continuar con la instalación de Harbor.

9 Compruebe que Harbor está instalado.

   a Seleccione el espacio de nombres de vSphere denominado *svc-harbor-domain-XXXX*.

   b Seleccione la pestaña **Red** y después **Servicios**.

   c Debería ver varios contenedores instalados para Harbor, cada uno de ellos es un servicio de tipo ClusterIP.

## Configurar DNS para Harbor

Deberá registrar un nombre de dominio y configurar un registro de DNS para Harbor.

1 Seleccione **Administración de cargas de trabajo > Espacios de nombres**.

2 Seleccione el espacio de nombres de **Contour**.

3 Seleccione **Red > Servicios**.

4 Registre la dirección IP externa para el servicio de entrada Envoy (por ejemplo, 10.197.154.71).

5 Registre el nombre de dominio de Harbor (FQDN) que especificó en la configuración de Harbor.

6 Cree un registro "A" de DNS mediante AWS Route 53 o un servicio similar.

## Iniciar sesión en Harbor

Una vez configurado el DNS de Harbor, inicie sesión.

1 Vaya al nombre de dominio que registró para Harbor.

2 Inicie sesión en el dominio con admin|contraseña de que especificó en la configuración de Harbor.

3 Una vez que haya iniciado sesión, cambie la contraseña a una más segura.

## Configurar Supervisor para que confíe en el registro de Harbor (opcional)

Los clústeres de TKG se configuran automáticamente para confiar en el servicio Supervisor de Harbor cuando el clúster de TKG y Harbor se implementan en el mismo Supervisor. Sin embargo, Supervisor NO se configura automáticamente para confiar en el servicio Supervisor de Harbor al crear pods de vSphere. Complete estos pasos para establecer la confianza entre Supervisor y el servicio de Harbor actualizando el configmap con el certificado de CA de Harbor.

1 En Harbor, vaya a **Administración > Configuración > Configuración del sistema**.

2 Descargue el certificado raíz del registro, que es un archivo denominado `ca.crt`.

3 Configure la variable de entorno KUBE_EDITOR.

   Consulte Configurar un editor de texto para Kubectl.

4 Inicie sesión en Supervisor mediante kubectl.

   Consulte Conectarse a Supervisor como usuario de vCenter Single Sign-On con kubectl.

5 Cambie el contexto al contexto de Supervisor (dirección IP).

6 Edite configmap/image-fetch-ca-bundle mediante el siguiente comando:

```
kubectl edit configmap image-fetcher-ca-bundle -n kube-system
```

7 Copie el contenido del archivo ca.crt de Harbor y anexe el configmap debajo del certificado existente (que es para Supervisor y no se debe cambiar). Guarde las ediciones realizadas en el archivo. Debería ver que Kubectl notifica "configmap/image-fetcher-ca-bundle edited".

## Configurar un cliente de Docker con el certificado de registro de Harbor

Para trabajar con imágenes de contenedor en un registro mediante Docker, agregue el certificado del registro al cliente de Docker. El certificado se utiliza para autenticar Docker durante el inicio de sesión en el registro.

Configure el cliente de Docker para que interactúe con un registro de contenedores (como el Registro de Harbor o Docker Hub). En este ejemplo, se supone que está utilizando servicio de supervisor de Harbor.

#### Requisitos previos

Esta tarea supone que se utiliza un host Linux (Ubuntu) en el que se instaló el daemon de Docker. Para instalar Docker Engine (daemon) en un host de Ubuntu, consulte https://docs.docker.com/engine/install/ubuntu/.

Para comprobar que Docker esté instalado y pueda extraer imágenes de Docker Hub, ejecute el siguiente comando:

```
docker run hello-world
```

Resultado esperado:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

**Nota** Estas instrucciones se comprueban mediante Ubuntu 20.04 y Docker 19.03.

#### Procedimiento

1 Inicie sesión en Registro de Harbor.

2 Seleccione **Administración > Configuración > Certificado raíz del registro**.

3 Haga clic en **Descargar** para descargar el certificado de Registro de Harbor denominado `ca.crt`.

   **Nota** De ser necesario, cambie el nombre del certificado a `ca.crt`.

4 Copie de forma segura el archivo `ca.crt` en el cliente de host de Docker.

5 En el host de Docker, cree una ruta de directorio para el registro privado utilizando la dirección IP de Harbor.

   ```
   /etc/docker/certs.d/IP-address-or-FQDN-of-harbor/
   ```

   Por ejemplo:

   ```
   mkdir /etc/docker/certs.d/10.179.145.77
   ```

6 Mueva `ca.crt` a este directorio.

   Por ejemplo:

   ```
    mv ca.crt /etc/docker/certs.d/10.179.145.77/ca.crt
   ```

7 Reinicie el daemon de Docker.

   ```
   sudo systemctl restart docker.service
   ```

8 Inicie sesión en el registro de Harbor integrado mediante su cliente de Docker.

   ```
   docker login https://10.179.145.77
   ```

Debería ver el siguiente mensaje:

```
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

# Insertar paquetes estándar en un registro de Harbor privado

Consulte este tema para insertar los paquetes estándar del registro de contenedor público en un registro de Harbor privado que se ejecute como un servicio de Supervisor.

## Requisitos previos

Complete los siguientes requisitos previos:

- Instalar el servicio Supervisor de Harbor.
- Configure un host de Ubuntu como cliente de Docker. Consulte Configurar un cliente de Docker con el certificado de registro de Harbor.
- Instale jq en el host Ubuntu. Consulte https://jqlang.github.io/jq/download/.

## Instalar la utilidad de paquetes de imágenes Carvel

Instale la utilidad `imgpkg` Carvel en el cliente Ubuntu en el que está instalado kubectl.

1 Instale `imgpkg`.

```
wget -O- https://carvel.dev/install.sh > install.sh
sudo bash install.sh
```

2 Compruebe la instalación.

```
imgpkg version
```

Resultado esperado:

```
imgpkg version 0.41.1
```

## Enumerar las imágenes disponibles para cada paquete estándar

Administrador de certificados

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/cert-manager
```

Contour con Envoy

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/contour
```

ExternalDNS

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/external-dns
```

Prometheus con Alertmanager

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/prometheus
```

Grafana

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/grafana
```

Fluent Bit

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/fluent-bit
```

Harbor

```
imgpkg tag list -i projects.registry.vmware.com/tkg/packages/standard/harbor
```

## Extraer las imágenes de paquete estándar del registro de VMware público

Extraiga los paquetes Tanzu Standard del registro de VMware público en https://projects.registry.vmware.com/. Actualice la versión para que coincida con la versión que desea extraer.

Administrador de certificados

```
docker pull projects.registry.vmware.com/tkg/packages/standard/cert-manager:v1.7.2_vmware.3-tkg.3
```

Contour con Envoy

```
docker pull projects.registry.vmware.com/tkg/packages/standard/contour:v1.23.5_vmware.1-tkg.1
```

ExternalDNS

```
docker pull projects.registry.vmware.com/tkg/packages/standard/external-dns:v0.12.2_vmware.5-tkg.1
```

Prometheus con Alertmanager

```
docker pull projects.registry.vmware.com/tkg/packages/standard/prometheus:v2.37.0_vmware.3-tkg.1
```

Grafana

```
docker pull projects.registry.vmware.com/tkg/packages/standard/grafana:v7.5.17_vmware.2-tkg.1
```

Fluent Bit

```
docker pull projects.registry.vmware.com/tkg/packages/standard/fluent-bit:v1.9.5_vmware.1-
tkg.2
```

Harbor

```
docker pull projects.registry.vmware.com/tkg/packages/standard/harbor:v2.7.1_vmware.1-tkg.1
```

## Crear un proyecto en el registro de Harbor privado

Cree un proyecto público en Harbor para alojar los paquetes de Tanzu.

1 Inicie sesión en el registro de Harbor privado.

2 En Harbor, seleccione **Proyectos > Nuevo proyecto**.

3 Cree un nuevo proyecto público denominado **tanzu-packages**.

## Etiquetar las imágenes de paquete estándar

Etiquete las imágenes utilizando la siguiente sintaxis.

```
docker tag SOURCE_IMAGE[:TAG] harbordomain.com/tanzu-packages/REPOSITORY[:TAG]
```

Donde:

- SOURCE_IMAGE[:TAG] es el nombre de la imagen extraída
- *harbordomain.com* es el nombre de DNS de su servidor de Harbor
- REPOSITORY[:TAG] es el nombre de la etiqueta de imagen

Administrador de certificados

```
docker tag projects.registry.vmware.com/tkg/packages/standard/cert-manager:v1.7.2_vmware.3-
tkg.3 harbordomain.com/tanzu-packages/cert-manager:v1.7.2
```

Contour con Envoy

```
docker tag projects.registry.vmware.com/tkg/packages/standard/contour:v1.23.5_vmware.1-tkg.1
harbordomain.com/tanzu-packages/contour:v1.23.5
```

ExternalDNS

```
docker tag projects.registry.vmware.com/tkg/packages/standard/external-dns:v0.12.2_vmware.5-
tkg.1 harbordomain.com/tanzu-packages/external-dns:v0.12.2
```

Prometheus con Alertmanager

```
docker tag projects.registry.vmware.com/tkg/packages/standard/prometheus:v2.37.0_vmware.3-
tkg.1 harbordomain.com/tanzu-packages/prometheus:v2.37.0
```

Grafana

```
docker tag projects.registry.vmware.com/tkg/packages/standard/grafana:v7.5.17_vmware.2-tkg.1
harbordomain.com/tanzu-packages/grafana:v7.5.17
```

Fluent Bit

```
docker tag projects.registry.vmware.com/tkg/packages/standard/fluent-bit:v1.9.5_vmware.1-
tkg.2 harbordomain.com/tanzu-packages/fluent-bit:v1.9.5
```

Harbor

```
docker tag projects.registry.vmware.com/tkg/packages/standard/harbor:v2.7.1_vmware.1-tkg.1
harbordomain.com/tanzu-packages/harbor:v2.7.1
```

## Insertar las imágenes de paquete estándar en el registro de Harbor privado

Inserte las imágenes utilizando la siguiente sintaxis.

```
docker push harbordomain.com/tanzu-packages/PACKAGE
```

Donde:

- *harbordomain.com* es el nombre de DNS de su servidor de Harbor
- *tanzu-packages* es el nombre del proyecto de Harbor
- *PACKAGE* es el nombre del paquete de Tanzu
- *vX.X.X* es la versión de etiqueta del paquete

Administrador de certificados

```
docker push harbordomain.com/tanzu-packages/cert-manager:v1.7.2
```

Contour con Envoy

```
docker push harbordomain.com/tanzu-packages/contour:v1.23.5
```

ExternalDNS

```
docker push harbordomain.com/tanzu-packages/external-dns:v0.12.2
```

Prometheus con Alertmanager

```
docker push harbordomain.com/tanzu-packages/prometheus:v2.37.0
```

Grafana

```
docker push harbordomain.com/tanzu-packages/grafana:v7.5.17
```

Fluent Bit

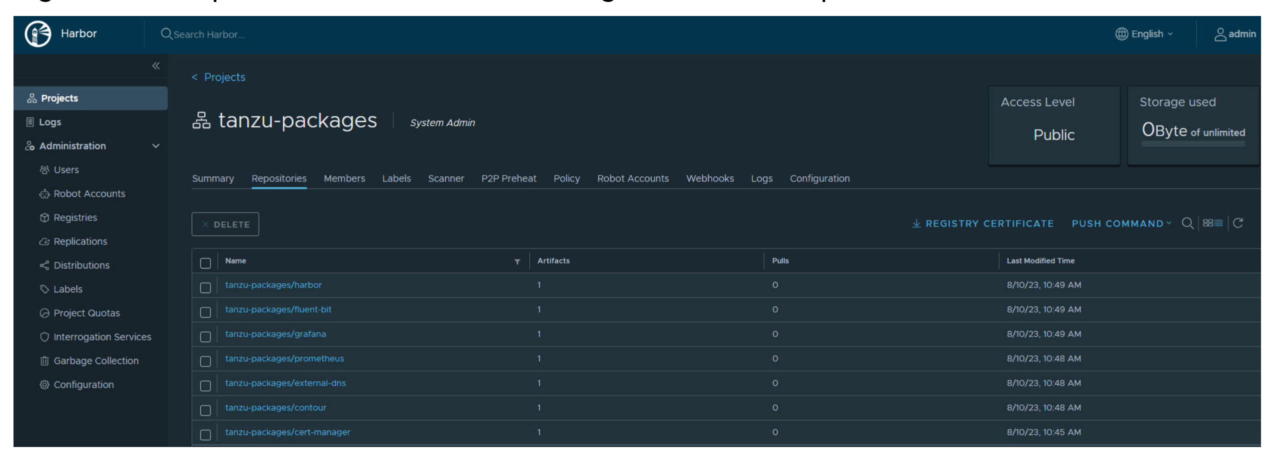
```
docker push harbordomain.com/tanzu-packages/fluent-bit:v1.9.5
```

Harbor

```
docker push harbordomain.com/tanzu-packages/harbor:v2.7.1
```

Después de insertar las imágenes en el registro de Harbor privado, compruebe que ve cada imagen en la interfaz web de Harbor.

**Figura 14-1. Paquetes Tanzu Standard en el registro de Harbor privado**




## Extraer una imagen de contenedor para la implementación

Para comprobar que las imágenes se encuentran en el registro de Harbor privado, extraiga una imagen desde allí con la siguiente sintaxis:

```
docker pull harbordomain.com/tanzu-packages/PACKAGE:TAG
```

Por ejemplo:

```
docker pull harbordomain.com/tanzu-packages/fluent-bit:v1.9.5
```

Puede utilizar cURL para invocar la API de Harbor y enumerar los paquetes. Para ello, descargue el certificado de Harbor desde la interfaz web de Harbor y después ejecute el siguiente comando:

```
curl -X 'GET' 'https://harbordomain.com/api/v2.0/projects/tanzu-packages/repositories?
page=1&page_size=-1' -H 'accept: application/json' --cacert ca.crt | jq '.[].name'
```

El comando devuelve los tanzu-packages que están disponibles.

```
"tanzu-packages/harbor"
"tanzu-packages/fluent-bit"
"tanzu-packages/grafana"
"tanzu-packages/prometheus"
"tanzu-packages/external-dns"
"tanzu-packages/contour"
"tanzu-packages/cert-manager"
```

# Crear instantáneas en un clúster de Servicio TKG

15

Los clústeres de Servicio TKG admiten capacidades de instantáneas y restauración de volúmenes. Como usuario de desarrollo y operaciones, puede proteger las cargas de trabajo en clústeres de TKG mediante la creación de instantáneas de volumen.

Puede utilizar una instantánea para aprovisionar un nuevo volumen, rellenado previamente con los datos de la instantánea.

## Requisitos previos

A fin de crear instantáneas de CSI para clústeres de Servicio TKG, el entorno debe cumplir con los siguientes requisitos previos.

- vSphere 8.0 Update 2 o una versión posterior.
- versión de Tanzu Kubernetes que admita instantáneas de CSI, que es TKr v1.26.5 o versiones posteriores para vSphere 8.0.2 o versiones posteriores. Consulte las Notas de la versión de las distintas versiones de VMware Tanzu Kubernetes.
- Versión de Supervisor compatible más reciente. Consulte las Notas de la versión de VMware vSphere with Tanzu 8.0.

## Requisitos

La funcionalidad de instantánea de CSI se ofrece como un paquete de TKG. Los requisitos para usar el paquete de instantáneas de CSI son los siguientes:

- Utilice el repositorio de paquetes estándar de TKG 2023.9.19 o versiones posteriores. Consulte Versiones del repositorio de paquetes Tanzu Standard.
- Instale el paquete Administrador de certificados. ConsulteInstalar y utilizar los paquetes de VMware Tanzu.
- Instale e implemente vsphere-pv-csi-webhook mediante la CLI de Tanzu. Consulte Instalar e implementar el webhook de PVCSI de vSphere.
- Instale e implemente external-csi-snapshot-webhook mediante la CLI de Tanzu. Consulte Instalar e implementar el webhook externo de instantáneas de CSI.

# Directrices y limitaciones

Cuando utilice la funcionalidad de instantáneas y restauración con clústeres de TKG, siga estas directrices.

- Solo los volúmenes de bloque admiten operaciones de instantáneas y restauración de volúmenes. No puede usar estas operaciones con volúmenes de archivos de vSphere.
- Cuando se crea una PVC a partir de VolumeSnapshot, debe residir en el mismo almacén de datos que el VolumeSnapshot original. De lo contrario, se produce el siguiente error en el aprovisionamiento de esa PVC:

  ```
  failed to provision volume with StorageClass <storage-class-name>: rpc error: code =
  Internal desc = failed to create volume. Error: failed to get the compatible datastore
  for create volume from snapshot <snapshot-name> with error: failed to find datastore with
  URL <datastore-url> from the input datastore list, <[datastore-list]>
  ```

  El almacén de datos de la PVC de destino que se crea a partir de VolumeSnapshot lo determina StorageClass en la definición de PVC. Asegúrese de que StorageClass de la PVC de destino y StorageClass de la PVC de origen original apunten al mismo almacén de datos, que es el almacén de datos de la PVC de origen. Esta regla también se aplica a los requisitos de topología en las definiciones de StorageClass. Los requisitos también deben apuntar al mismo almacén de datos común. Cualquier requisito de topología en conflicto provocará el mismo error que el mostrado anteriormente.
- No puede eliminar ni expandir un volumen que contenga instantáneas asociadas. Elimine todas las instantáneas para expandir o eliminar el volumen de origen.
- Cuando cree un volumen a partir de una instantánea, asegúrese de que el tamaño del volumen coincida con el tamaño de la instantánea.
- No se admite la supervisión de la cuota de almacenamiento para instantáneas.
- No puede configurar el número máximo de instantáneas por volumen en la configuración de vSphere. Para obtener un mejor rendimiento, utilice de dos a tres instantáneas por disco virtual. Para obtener más información, consulte Prácticas recomendadas para usar instantáneas de VMware en el entorno de vSphere.

  Para vSAN ESA, utilice un máximo de 32 instantáneas por volumen. Para obtener más información sobre vSAN ESA, consulte vSAN Express Storage Architecture.

Lea los siguientes temas a continuación:

- Instalar e implementar el webhook externo de instantáneas de CSI
- Instalar e implementar el webhook de PVCSI de vSphere
- Crear instantáneas en un clúster de Servicio TKG

# Instalar e implementar el webhook externo de instantáneas de CSI

Para poder utilizar la tecnología de instantáneas en los clústeres de Servicio TKG, debe instalar e implementar un webhook externo de instantáneas de CSI en un clúster de TKG. El webhook externo de instantáneas CSI es un componente de código abierto con una llamada de retorno HTTP que responde a las solicitudes de admisión. Es responsable de la validación de los objetos de instantáneas de volumen.

El webhook externo de instantáneas de CSI se instala automáticamente en Supervisor. Este tema solo se aplica a los clústeres de Servicio TKG.

## Requisitos previos

- Tiene un Supervisor en ejecución.
- Ha instalado la CLI de Tanzu y kubectl. Para obtener información, consulte Instalar las herramientas de CLI para clústeres de Servicio TKG.
- Inició sesión en el clúster de TKG. Para obtener información, consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

## Preparar un clúster de Servicio TKG para la instalación de un webhook externo de instantáneas de CSI

Siga estos pasos para instalar un webhook externo de instantáneas de CSI en un clúster de Servicio TKG.

### Procedimiento

1 Obtenga las credenciales de administrador del clúster de TKG en el que desea implementar el webhook externo de instantáneas CSI.

```
tanzu cluster kubeconfig get my-cluster --admin
```

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
kubectl config use-context my-cluster-admin@my-cluster
```

3 Si el clúster no tiene un repositorio de paquetes, como el repositorio tanzu-standard, instale uno.

   Puede omitir este paso si el clúster de destino es un clúster heredado basado en planes. Para los clústeres basados en planes, el repositorio de paquetes tanzu-standard se habilita automáticamente en el espacio de nombres tanzu-package-repo-global.

```
tanzu package repository add PACKAGE-REPO-NAME --url PACKAGE-REPO-ENDPOINT --namespace tkg-system
```

   - PACKAGE-REPO-NAME es el nombre del repositorio de paquetes, como tanzu-standard, o el nombre de un registro de imágenes privado configurado con variables ADDITIONAL_IMAGE_REGISTRY.
   - PACKAGE-REPO-ENDPOINT es la URL del repositorio de paquetes.

4 Instale el archivo administrador de certificados si aún no lo ha hecho.

   Para obtener información, consulte Instalar administrador de certificados.

#### Resultados

Ahora puede implementar el webhook externo de instantáneas CSI.

## Implementar webhook externo de instantáneas de CSI

Siga estos pasos para implementar el webhook externo de instantáneas de CSI en un clúster de Servicio TKG.

#### Procedimiento

1 Confirme que el paquete de webhook externo de instantáneas CSI está disponible en el clúster.

```
tanzu package available list -A
```

   Si el paquete no está disponible, asegúrese de que el repositorio de paquetes que contiene el paquete de webhook externo de instantáneas CSI está instalado correctamente. Para obtener instrucciones, consulte el paso 3 de Preparar un clúster de Servicio TKG para la instalación de un webhook externo de instantáneas de CSI.

2 Obtenga la versión del paquete disponible.

```
tanzu package available list external-csi-snapshot-webhook.tanzu.vmware.com -A
```

3 Instale el paquete con la versión adecuada disponible.

```
tanzu package install external-csi-snapshot-webhook --package external-csi-snapshot-webhook.tanzu.vmware.com --version AVAILABLE-PACKAGE-VERSION --namespace kube-system
```

   AVAILABLE-PACKAGE-VERSION especifica la versión del paquete que obtuvo en el paso 2.

4 Confirme que está instalado el paquete de webhook externo de instantáneas CSI.

```
tanzu package installed list -A
```

Para ver más detalles sobre el paquete, también puede ejecutar el siguiente comando:

```
tanzu package installed get external-csi-snapshot-webhook --namespace kube-system
```

5 Confirme que la aplicación external-csi-snapshot-webhook se ha conciliado correctamente en su TARGET-NAMESPACE.

```
kubectl get apps -A
```

Si el estado no es `Reconcile Succeeded`, consulte los detalles del estado completo de la aplicación external-csi-snapshot-webhook. Ver el estado completo puede ayudarle a solucionar el problema.

```
kubectl get app external-csi-snapshot-webhook --namespace kube-system -o yaml
```

Si la solución de problemas no le ayuda a resolver el problema, desinstale el paquete con el siguiente comando antes de volver a instalarlo.

```
tanzu package installed delete external-csi-snapshot-webhook --namespace kube-system
```

6 Enumere todos los pods del clúster para confirmar que el external-csi-snapshot-webhook se está ejecutando.

```
kubectl get pods -A
```

Compruebe que los pods external-csi-snapshot-webhook se han creado en el espacio de nombres kube-system.

# Instalar e implementar el webhook de PVCSI de vSphere

Instale e implemente el webhook de PVCSI de vSphere en un clúster de Servicio TKG. El webhook de PVCSI de vSphere es un componente con llamada de retorno que responde a las solicitudes de admisión de CSI. Se encarga de la validación de los objetos de Kubernetes, como las notificaciones de volumen persistente, los volúmenes persistentes, las clases de almacenamiento, etc.

El webhook de PVCSI de vSphere se instala de forma automática en Supervisor. Este tema solo se aplica a los clústeres de Servicio TKG.

## Requisitos previos

- Tiene un Supervisor en ejecución.
- Ha instalado la CLI de Tanzu y kubectl. Para obtener información, consulte Instalar las herramientas de CLI para clústeres de Servicio TKG.

- Inició sesión en el clúster de TKG. Para obtener información, consulte Conectarse a clústeres de Servicio TKG mediante la autenticación de vCenter SSO.

## Preparar un clúster de Servicio TKG para la instalación del webhook de PVCSI de vSphere

Siga estos pasos a fin de preparar el clúster de Servicio TKG para la instalación del webhook de PVCSI de vSphere.

#### Procedimiento

1 Obtenga las credenciales de administrador del clúster de TKG en el que desea implementar el webhook de PVCSI de vSphere.

```
tanzu cluster kubeconfig get my-cluster --admin
```

2 Cambie el contexto al espacio de nombres de vSphere donde se aprovisiona el clúster de TKG de destino.

```
kubectl config use-context my-cluster-admin@my-cluster
```

3 Si el clúster no tiene un repositorio de paquetes con el paquete vsphere-pv-csi-webhook instalado, como el repositorio tanzu-standard, instale uno.

   Puede omitir este paso si el clúster de destino es un clúster heredado basado en planes. Para los clústeres basados en planes, el repositorio de paquetes tanzu-standard se habilita automáticamente en el espacio de nombres tanzu-package-repo-global.

```
tanzu package repository add PACKAGE-REPO-NAME --url PACKAGE-REPO-ENDPOINT --namespace tkg-
system
```

   - PACKAGE-REPO-NAME es el nombre del repositorio de paquetes, como tanzu-standard, o el nombre de un registro de imágenes privado configurado con variables ADDITIONAL_IMAGE_REGISTRY.
   - PACKAGE-REPO-ENDPOINT es la URL del repositorio de paquetes.

4 Instale el archivo administrador de certificados si aún no lo ha hecho.

   Para obtener información, consulte Instalar administrador de certificados.

#### Resultados

Ahora puede implementar el webhook de PVCSI de vSphere.

## Implementar el webhook de PVCSI de vSphere

Siga estos pasos para implementar el webhook de PVCSI de vSphere en un clúster de Servicio TKG.

Procedimiento

1 Confirme que el paquete del webhook de PVCSI de vSphere esté disponible en el clúster.

```
tanzu package available list -A
```

Si el paquete no está disponible, asegúrese de que el repositorio de paquetes que contiene el paquete del webhook de PVCSI de vSphere requerido está instalado correctamente. Para obtener instrucciones, consulte el paso 3 de Preparar un clúster de Servicio TKG para la instalación del webhook de PVCSI de vSphere.

2 Obtenga la versión del paquete disponible.

```
tanzu package available list vsphere-pv-csi-webhook.tanzu.vmware.com -A
```

3 Instale el paquete con la versión adecuada disponible.

```
tanzu package install vsphere-pv-csi-webhook --package vsphere-pv-csi-
webhook.tanzu.vmware.com --version AVAILABLE-PACKAGE-VERSION --namespace TARGET-NAMESPACE
```

- TARGET-NAMESPACE especifica el espacio de nombres en el que desea instalar el paquete vsphere-pv-csi-webhook.

  **Nota** TARGET-NAMESPACE debe ser el mismo que el espacio de nombres en el que está instalado el paquete vsphere-pv-csi.

  Si no especifica la marca `--namespace`, la CLI de Tanzu instala el paquete y sus recursos en el espacio de nombres predeterminado, como `vmware-system-csi` para el paquete vsphere-pv-csi-webhook. El espacio de nombres especificado ya debe existir (por ejemplo, al ejecutar `kubectl create namespace vmware-system-csi`).

- AVAILABLE-PACKAGE-VERSION especifica la versión del paquete que obtuvo en el paso 2.

4 Confirme que se haya instalado el paquete del webhook de PVCSI de vSphere.

```
tanzu package installed list -A
```

Para ver más detalles sobre el paquete, también puede ejecutar el siguiente comando:

```
tanzu package installed get vsphere-pv-csi-webhook --namespace TARGET-NAMESPACE
```

5 Confirme que la aplicación vsphere-pv-csi-webhook se ha conciliado correctamente en su TARGET-NAMESPACE.

```
kubectl get apps -A
```

Si el estado no es `Reconcile Succeeded`, consulte los detalles del estado completo de la aplicación vsphere-pv-csi-webhook. Ver el estado completo puede ayudarle a solucionar el problema.

```
kubectl get app vsphere-pv-csi-webhook --namespace TARGET-NAMESPACE -o yaml
```

Si la solución de problemas no le ayuda a resolver el problema, desinstale el paquete con el siguiente comando antes de volver a instalarlo.

```
tanzu package installed delete vsphere-pv-csi-webhook --namespace TARGET-NAMESPACE
```

6 Enumere todos los pods del clúster para confirmar que vsphere-pv-csi-webhook se está ejecutando.

```
kubectl get pods -A
```

Compruebe que han creado los pods vsphere-pv-csi-webhook en vmware-system-csi O EN TARGET-NAMESPACE.

# Crear instantáneas en un clúster de Servicio TKG

Puede aprovisionar una instantánea de forma dinámica o crear una instantánea de volumen aprovisionada previamente de un volumen de bloque en un clúster de Servicio TKG. También puede restaurar una instantánea existente.

**Nota** No debe crear clases de instantáneas de volumen y utilizarlas para crear instantáneas de volumen. Utilice solo una clase de instantánea de volumen preexistente. Las clases de instantáneas de volúmenes preexistentes solo admiten la deletionPolicy Eliminar. No se admite la creación de clases de instantáneas de volumen con la directiva deletionPolicy Conservar.

## Crear una instantánea aprovisionada dinámicamente en un clúster de Servicio TKG

Aprovisione dinámicamente una instantánea en un clúster de Servicio TKG.

### Procedimiento

1 Compruebe que StorageClass está presente.

```
$ kubectl get sc
NAME                             PROVISIONER              RECLAIMPOLICY
VOLUMEBINDINGMODE       ALLOWVOLUMEEXPANSION   AGE
gc-storage-profile               csi.vsphere.vmware.com   Delete
Immediate               true                   11d
gc-storage-profile-latebinding   csi.vsphere.vmware.com   Delete
WaitForFirstConsumer    true                   11d
```

2 Cree una PVC usando la StorageClass del paso 1.

Utilice el siguiente YAML como ejemplo.

```
$ cat example-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-raw-block-pvc
spec:
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: gc-storage-profile
```

```
$ kubectl apply -f example-pvc.yaml
```

```
$ kubectl get pvc
NAME                    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS         AGE
example-raw-block-pvc   Bound    pvc-4c0c030d-25ac-4520-9a04-7aa9361dfcfc   1Gi
RWO            gc-storage-profile   2m1s
```

3 Compruebe que VolumeSnapshotClass esté disponible.

```
$ kubectl get volumesnapshotclass
NAME                          DRIVER                   DELETIONPOLICY   AGE
volumesnapshotclass-delete    csi.vsphere.vmware.com   Delete           11d
```

4 Cree un VolumeSnapshot usando el valor de VolumeSnapshotClass que obtuvo en el paso 3.

```
$cat example-snapshot.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: example-raw-block-snapshot
spec:
  volumeSnapshotClassName: volumesnapshotclass-delete
  source:
    persistentVolumeClaimName: example-raw-block-pvc
```

```
$ kubectl apply -f example-snapshot.yaml
```

```
$ kubectl get volumesnapshot
NAME                          READYTOUSE   SOURCEPVC               SOURCESNAPSHOTCONTENT
RESTORESIZE   SNAPSHOTCLASS                 SNAPSHOTCONTENT
                            CREATIONTIME   AGE
```

```
example-raw-block-snapshot   true         example-raw-block-pvc
1Gi           volumesnapshotclass-delete   snapcontent-ae019c16-
b07c-4a92-868b-029babd641d3   6s             6s
```

## Crear una instantánea aprovisionada previamente en un clúster de Servicio TKG

Cree una instantánea de volumen aprovisionada previamente de cualquier volumen de bloque (PVC) en un clúster de Servicio TKG mediante una instantánea de volumen del mismo volumen de bloque (PVC) de Supervisor.

Siga estos pasos para crear estáticamente una instantánea de volumen en un nuevo clúster de TKG utilizando la información de la instantánea subyacente de sobra en Supervisor.

**Nota** No se pueden crear instantáneas de volumen directamente en un Supervisor.

### Requisitos previos

- Familiarícese con la información sobre la creación de instantáneas de Kubernetes. Para obtener más información, consulte la página Instantáneas de volumen en la documentación de Kubernetes.
- Asegúrese de que la instantánea de volumen cumple las siguientes condiciones:
  - La instantánea de volumen está presente en el mismo espacio de nombres en el que reside el PVC de origen.
  - La instantánea de volumen está presente en el mismo espacio de nombres en el que reside el clúster de TKG.

También puede reutilizar en un nuevo clúster de TKG una instantánea de volumen que ya no la necesite otro clúster de TKG en el mismo espacio de nombres. Para ello, cambie `deletionPolicy` de `VolumeSnapshotContent` en el clúster de TKG original a `Retain` y después elimine la `VolumeSnapshot` correspondiente, así como los objetos `VolumeSnapshotContent`.

### Procedimiento

1 Anote el nombre del objeto `VolumeSnapshot` original en el Supervisor.

  Si reutiliza la instantánea de volumen de un clúster de TKG antiguo, también puede recuperar el nombre de `VolumeSnapshot` de Supervisor del `snapshotHandle` del objeto `VolumeSnapshotContent` anterior existente en el clúster de TKG anterior.

2 Cree un objeto `VolumeSnapshotContent`.

  En el archivo YAML, especifique el valor del siguiente elemento:

Para `snapshotHandle`, introduzca el nombre de `VolumeSnapshot` de Supervisor que obtuvo en el paso 1.

---

**Nota** Nota: Si vuelve a utilizar una instantánea de volumen de otro clúster de TKG, elimine los objetos `VolumeSnapshot` y `VolumeSnapshotContent` del clúster de TKG anterior, con `deletionPolicy` establecido en Conservar, antes de crear un `VolumeSnapshotContent` en el nuevo clúster de TKG.

---

Utilice el siguiente manifiesto de YAML como ejemplo.

```
--------
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
 name: static-tkg-block-snapshotcontent
spec:
 deletionPolicy: Delete
 driver: csi.vsphere.vmware.com
 source:
  snapshotHandle: "supervisor-block-volumeSnapshot-name" # Enter the VolumeSnapshot name 
from the Supervisor.
 volumeSnapshotRef:
  name: static-tkg-block-snapshot
  namespace: "supervisor-tkg-namespace" # Enter the namespace of Tanzu Kubernetes Grid 
Cluster.
---------
```

3 Cree un `VolumeSnapshot` que concuerde con el objeto `VolumeSnapshotContent` que creó en el paso 2.

```
---------
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: static-tkg-block-snapshot
spec:
 source:
  volumeSnapshotContentName: static-tkg-block-snapshotcontent
---------
```

4 Compruebe que la `VolumeSnapshot` que creó en el paso 3 esté marcada con ReadyToUse como "true".

```
kubecti getvolumesnapshot static-tkg-block-snapshot
NAME                       READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT
RESTORESIZE SNAPSHOTCLASS SNAPSHOTCONTENT                  CREATIONTIME AGE
static-tkg-block-snapshot  true                 static-tkg-biock-snapshotcontent
5Gi                       static-tkg-block-snapsnotcontent 76m          22m
```

# Restaurar una instantánea de volumen en un clúster de Servicio TKG

Restaure una instantánea de volumen que ya se haya creado.

### Procedimiento

1 Compruebe que la instantánea del volumen que desea restaurar esté disponible en el clúster de TKG.

```
$ kubectl get volumesnapshot
NAME                          READYTOUSE   SOURCEPVC
SOURCESNAPSHOTCONTENT   RESTORESIZE   SNAPSHOTCLASS
SNAPSHOTCONTENT                                    CREATIONTIME   AGE
example-raw-block-snapshot   true         example-raw-block-pvc
1Gi           volumesnapshotclass-delete   snapcontent-ae019c16-
b07c-4a92-868b-029babd641d3   2m36s          2m36s
```

2 Cree una PVC a partir de la instantánea de volumen.

```
$ cat example-restore.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-raw-block-restore
spec:
  storageClassName: gc-storage-profile
  dataSource:
    name: example-raw-block-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  volumeMode: Block
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

```
$ kubectl apply -f example-restore.yaml
```

```
$ kubectl get pvc
NAME                        STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS         AGE
example-raw-block-pvc       Bound    pvc-4c0c030d-25ac-4520-9a04-7aa9361dfcfc   1Gi
RWO            gc-storage-profile   11m
example-raw-block-restore   Bound    pvc-96eaab16-9ec1-446a-9392-e86d13c9b2e2   1Gi
RWO            gc-storage-profile   2m8s
```

# Administrar el almacenamiento para clústeres de Servicio TKG

16

En esta sección, se proporciona información para la administración del almacenamiento de los clústeres de Servicio TKG.

Lea los siguientes temas a continuación:



## Conceptos de almacenamiento para clústeres del Servicio TKG

Las cargas de trabajo de clústeres de TKG pueden requerir almacenamiento persistente. Consulte en la información de este tema los conceptos y consideraciones sobre almacenamiento de vSphere para clústeres del Servicio TKG.

### Directivas de almacenamiento de vSphere para clústeres del Servicio TKG

Para proporcionar recursos de almacenamiento persistentes a los clústeres del Servicio TKG, un administrador de vSphere configura directivas de almacenamiento de vSphere que describen distintos requisitos de almacenamiento. A continuación, el administrador agrega una o varias directivas de almacenamiento al espacio de nombres de vSphere en el que se implementarán los clústeres de TKG. Las directivas de almacenamiento asignadas a un espacio de nombres de vSphere determinan cómo se colocan los nodos y las cargas de trabajo del clúster de TKG en el entorno de almacenamiento de vSphere y determinan a qué almacenes de datos pueden acceder los clústeres de TKG y cuáles pueden utilizar para almacenamiento persistente.

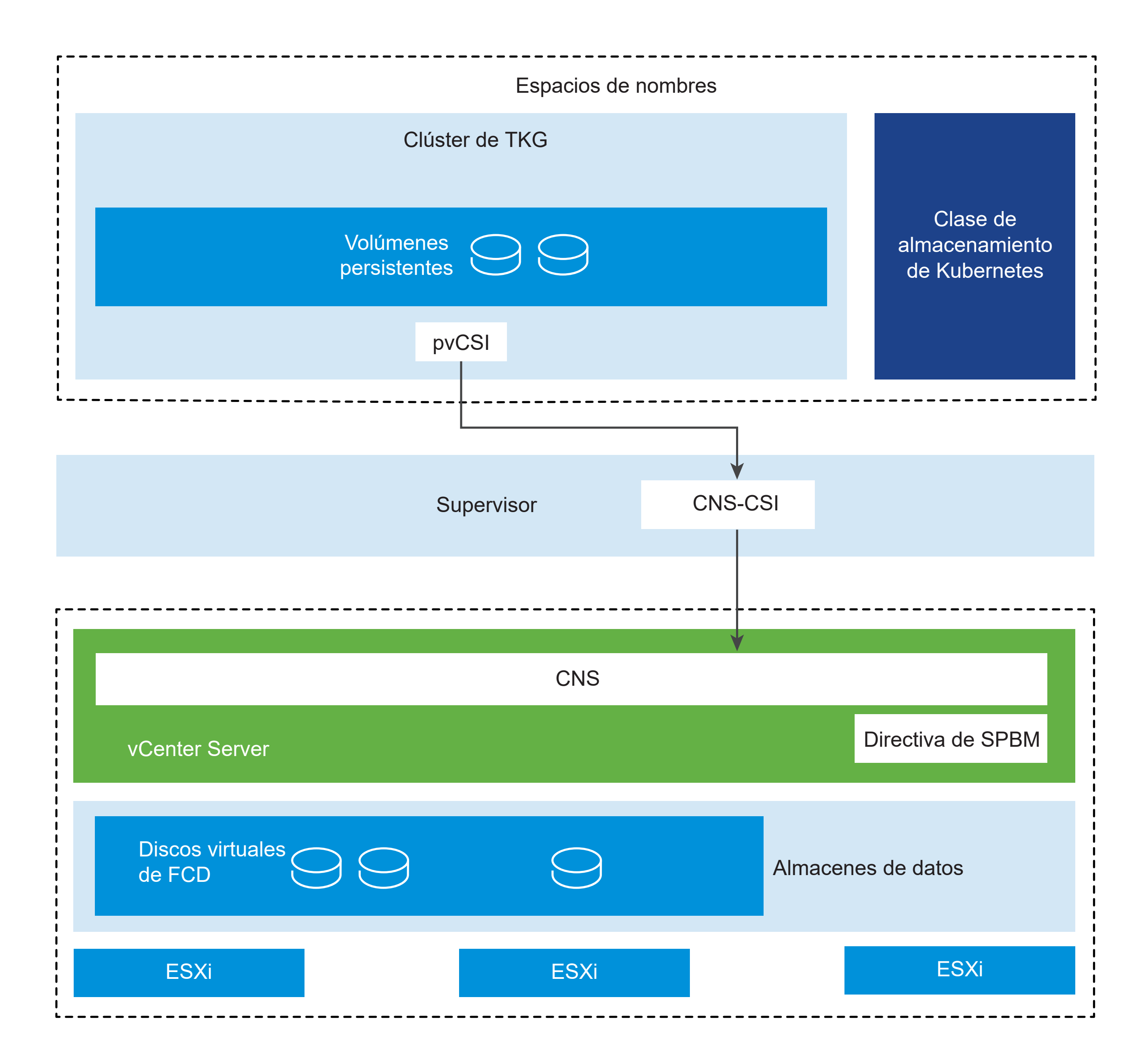
Cuando se asigna una directiva de almacenamiento de vSphere a un espacio de nombres de vSphere, el sistema crea clases de almacenamiento de Kubernetes correspondientes para ese espacio de nombres de vSphere. Esas clases de almacenamiento de Kubernetes correspondientes se propagan a los clústeres de TKG aprovisionados en ese espacio de nombres de vSphere.

En el clúster de TKG, cada clase de almacenamiento tiene dos ediciones, una con el modo de enlace `Immediate` y otra con el modo de enlace `WaitForFirstConsumer`. La edición que se elegirá depende de los requisitos que usted establezca. Consulte Ediciones de clase de almacenamiento para clústeres de Servicio TKG.

## Cómo se integran los clústeres de Servicio TKG con el almacenamiento de vSphere

Para integrarse con el Supervisor y el almacenamiento de vSphere, los clústeres de TKG usan Paravirtual CSI (pvCSI).

pvCSI es la versión del controlador de vSphere CNS-CSI modificada para los clústeres de TKG. pvCSI reside en el clúster de TKG y es responsable de todas las solicitudes relacionadas con el almacenamiento que se originan en el clúster de TKG. Las solicitudes se envían a CNS-CSI, que a su turno las propaga a CNS en vCenter Server. Como resultado, pvCSI no tiene comunicación directa con el componente de CNS, sino que depende del CNS-CSI para las operaciones de aprovisionamiento de almacenamiento. A diferencia de CNS-CSI, pvCSI no requiere credenciales de infraestructura. Se configura con una cuenta de servicio en el espacio de nombres de vSphere.

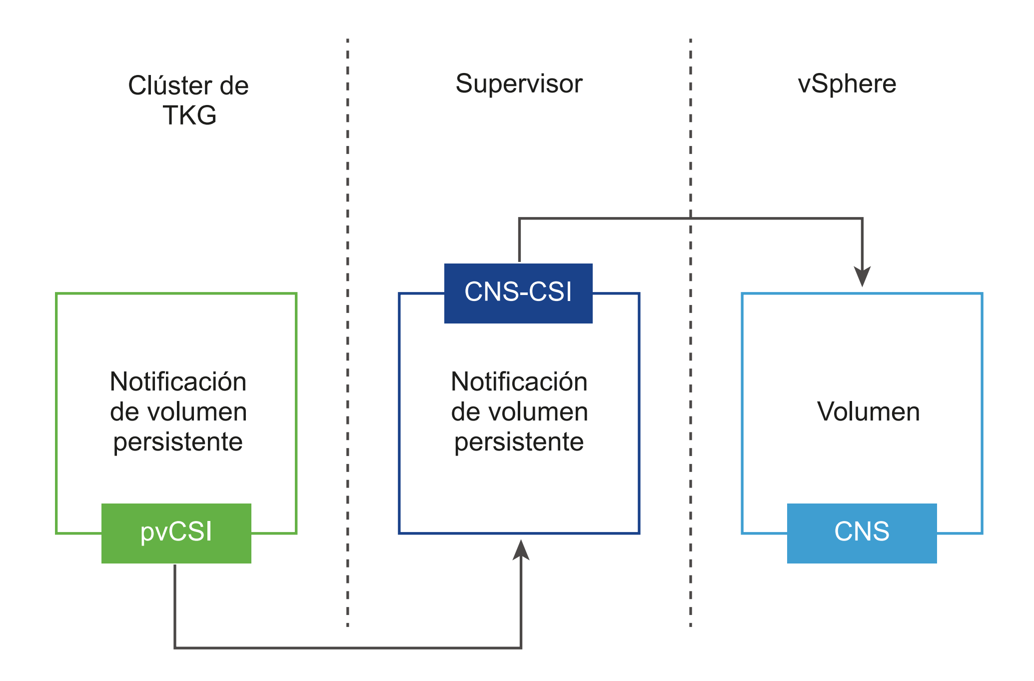



## Cómo se crea un volumen persistente

El diagrama muestra cómo interactúan los diferentes componentes para las operaciones relacionadas con el almacenamiento dentro de un clúster de TKG, como la creación de una notificación de volumen persistente (Persistent Volume Claim, PVC).

El ingeniero DevOps crea una PVC mediante kubectl en el clúster de TKG. Esta acción genera una PVC coincidente en el Supervisor y activa el CNS-CSI que invoca a la API de creación de volúmenes de CNS.

Después de crear correctamente un volumen, la operación se propaga de vuelta a través del Supervisor al clúster de TKG. Los usuarios del clúster pueden ver el volumen persistente y la notificación de volumen persistente en el estado enlazado del Supervisor. Además, también verán el volumen persistente y la notificación de volumen persistente en el estado enlazado del clúster de TKG.




# Consideraciones para usar montajes de volumen de nodo

Puede aprovisionar un clúster de servicio TKG con uno o varios montajes de volumen de nodo. Antes de hacerlo, tenga en cuenta las siguientes consideraciones importantes.

## Consideraciones sobre el montaje de volumen de nodo

En la tabla, se resumen los puntos de montaje de volumen de nodo para los clústeres de servicio TKG. Consulte el siguiente artículo de la base de conocimientos para obtener detalles completos sobre las limitaciones de montaje de volumen de nodo: https://kb.vmware.com/s/article/92153.

Consulte también el siguiente tema para obtener información sobre cómo cambiar un montaje de volumen después de aprovisionar un clúster: Escalar manualmente un clúster mediante Kubectl.

| Montaje de volumen | Nodo | Soporte | Desc |
|---|---|---|---|
| `/var/lib/containerd` | Trabajador | Completo | Aumenta el tamaño disponible para el almacenamiento en caché de imágenes de contenedor. |
| `/var/lib/kubelet` | Trabajador | Completo | Aumenta el tamaño disponible para el almacenamiento de contenedores efímeros. |

| Montaje de volumen | Nodo | Soporte | Desc |
|---|---|---|---|
| `/var/lib/etcd` | Plano de control | Ninguno | No aumenta el tamaño de etcd, porque tiene un límite estricto de 2 GB. También puede evitar la creación de clústeres debido a tiempos de espera de PVC. |
| `/` (raíz)<br>`/var`<br>`/var/lib`<br>`/etc` | Plano de control<br>Trabajador | Ninguno | No se admite ningún directorio utilizado en los procesos principales del sistema, incluidos los directorios que aparecen en la lista y cualquier otro (la lista no es exhaustiva). |

# Crear una directiva de almacenamiento de vSphere para clústeres de Servicio TKG

La directiva de almacenamiento de vSphere que se asigna a un espacio de nombres de vSphere se convierte en una clase de almacenamiento de Kubernetes. Utilice esta clase de almacenamiento para controlar cómo se colocan los nodos del clúster de TKG y los volúmenes persistentes dentro de almacenes de datos de vSphere. Una directiva de almacenamiento de vSphere para zonas de vSphere debe ser compatible con el almacenamiento en todos los clústeres de vSphere que comprenden la topología zonal.

## Definir una directiva de almacenamiento de vSphere para una zona única de Supervisor

Complete los pasos para crear una clase de almacenamiento para una zona única de Supervisor.

1 Con vSphere Client, seleccione **Directivas y perfiles**.

2 Seleccione **Directivas de almacenamiento de máquina virtual > Crear**.

3 Seleccione el vCenter Server.

4 Asigne un nombre descriptivo a la directiva de almacenamiento, como *TKG2-cluster-storage-class*

  Se utiliza el mismo nombre para la clase de almacenamiento que se crea.

5 Para la opción **Estructura de directiva** de almacenamiento, seleccione **Habilitar reglas para el almacenamiento "VMFS"**.

6 Para las reglas de VMFS, seleccione **Ahorrar espacio cuando sea posible**.

7 Para la compatibilidad de almacenamiento, revise y haga clic en **Siguiente**.

8 Haga clic en **Finalizar** para completar la creación de la directiva de almacenamiento.

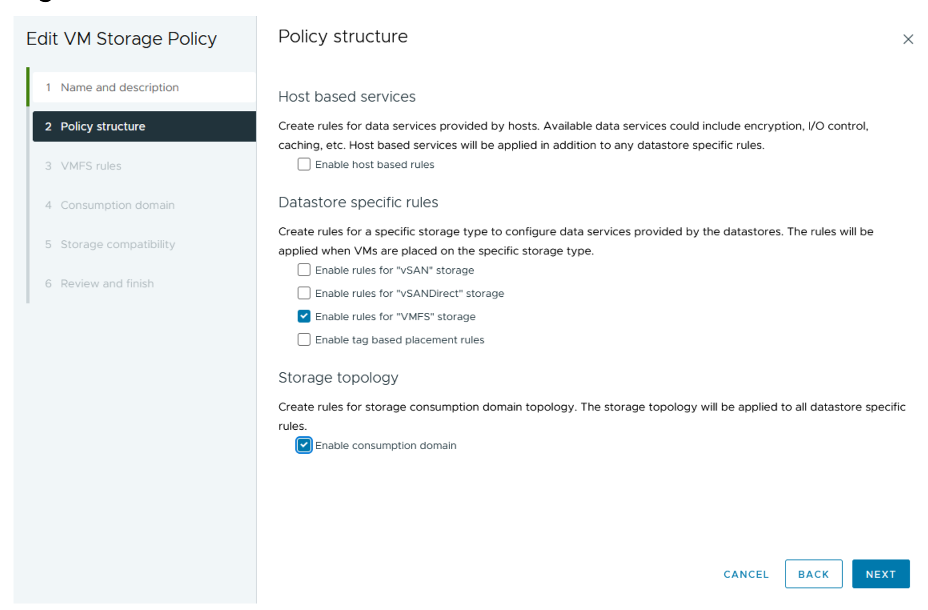
Para asignar la directiva de almacenamiento a un espacio de nombres de vSphere, consulte Configurar un espacio de nombres de vSphere para clústeres de Servicio TKG.

# Definir una directiva de almacenamiento de vSphere para Supervisor de tres zonas

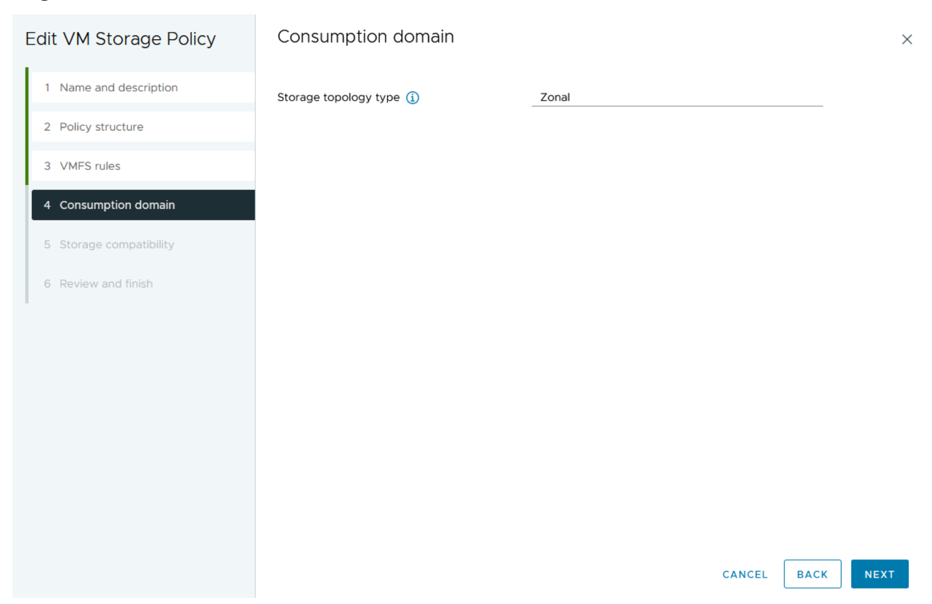
Si utiliza Supervisor implementado en tres zonas, complete los mismos pasos que se describen anteriormente, con las siguientes adiciones:

- Para la **estructura de la directiva** de almacenamiento, seleccione **Tipo de almacenamiento > Habilitar dominio de consumo**.
- Para **Dominio de consumo**, especifique **Zonal** como el tipo de directiva de almacenamiento.

### Figura 16-1. Clase de almacenamiento zonal 1 de 2




### Figura 16-2. Clase de almacenamiento zonal 2 de 2



## Creación avanzada de directivas de almacenamiento de vSphere

En función de su entorno, es posible que deba tener opciones de configuración adicionales para vSAN o vVols. Si utiliza VMFS y NFS, la directiva de almacenamiento de vSphere incluirá etiquetas.

Para crear tipos de directiva de almacenamiento más avanzados, como el almacenamiento basado en etiquetas, que se puedan aplicar a un espacio de nombres de vSphere para usarlo con clústeres de TKG, consulte *Instalar y configurar el plano de control de IaaS de vSphere*.

# Aprovisionar un volumen persistente dinámico para una aplicación con estado en el clúster de Servicio TKG

Las aplicaciones con estado (por ejemplo, bases de datos) guardan datos entre sesiones y requieren almacenamiento persistente para almacenar los datos. Estos datos que se conservan se denominan estado de la aplicación. Posteriormente, puede recuperarlos y utilizarlos en la siguiente sesión. Kubernetes ofrece volúmenes persistentes como objetos que pueden conservar su estado y sus datos.

En el entorno de vSphere, los objetos de volúmenes persistentes se respaldan con discos virtuales que residen en almacenes de datos. Los almacenes de datos se representan a través de directivas de almacenamiento. Después de que el administrador de vSphere crea una directiva de almacenamiento (por ejemplo, **`gold`**) y la asigna a un espacio de nombres en Supervisor, la directiva de almacenamiento se muestra como una clase de almacenamiento de Kubernetes coincidente en espacio de nombres de vSphere y en todos los clústeres de TKG disponibles.

Como ingeniero de desarrollo y operaciones, puede utilizar la clase de almacenamiento en sus especificaciones de notificación de volúmenes persistentes. Posteriormente, puede implementar una aplicación que utilice almacenamiento de la notificación de volumen persistente. En este ejemplo, el volumen persistente de la aplicación se crea de forma dinámica.

#### Requisitos previos

Asegúrese de que el administrador de vSphere haya creado una directiva de almacenamiento adecuada y haya asignado la directiva al espacio de nombres.

#### Procedimiento

1 Acceda al espacio de nombres en el entorno de Kubernetes de vSphere.

2 Compruebe que las clases de almacenamiento se encuentren disponibles.

3 Cree una notificación de volumen persistente (Persistent Volume Claim, PVC).

a Cree un archivo YAML que contenga la configuración de notificación de volumen persistente.

En este ejemplo, el archivo hace referencia a la clase de almacenamiento **gold**.

Para aprovisionar un volumen persistente ReadWriteMany, establezca `accessModes` en `ReadWriteMany`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 3Gi
```

b Aplique la notificación de volumen persistente al clúster de TKG.

```
kubectl apply -f pvc_name.yaml
```

Este comando crea de forma dinámica un volumen persistente de Kubernetes y un volumen de vSphere con un disco virtual de respaldo que cumple los requisitos de almacenamiento de la notificación.

c Compruebe el estado de la notificación de volumen persistente.

```
kubectl get pvc my-pvc
```

El resultado muestra que el volumen está enlazado a la notificación de volumen persistente.

```
NAME     STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc    2Gi        RWO           gold           30s
```

4 Cree un pod que monte el volumen persistente.

a Cree un archivo YAML que incluya el volumen persistente.

El archivo contiene estos parámetros.

```
...
volumes:
    - name: my-pvc
      persistentVolumeClaim:
        claimName: my-pvc
```

b Implemente el pod desde el archivo YAML.

```
kubectl create -f pv_pod_name.yaml
```

c Compruebe que se haya creado el pod.

```
kubectl get pod
```

```
NAME        READY   STATUS    RESTARTS   AGE
pod_name    1/1     Ready     0          40s
```

#### Resultados

El pod que configuró utilizará el almacenamiento persistente que se describe en la notificación de volumen persistente.

## Aprovisionamiento de un volumen persistente estático en un clúster de Servicio TKG

Puede crear de forma estática un volumen de bloques en un clúster de Servicio TKG mediante una notificación de volumen persistente (persistent volume claim, PVC) sin utilizar desde Supervisor.

La PVC debe cumplir las siguientes condiciones:

- La PVC debe estar presente en el mismo espacio de nombres en el que reside el clúster de TKG.
- La PVC aún no debe estar asociada a un pod en ningún clúster de TKG ni a un pod de vSphere en Supervisor.

Mediante el aprovisionamiento estático, también puede reutilizar en un clúster de TKG nuevo una PVC que otro clúster de TKG ya no necesite. Para ello, cambie la `Reclaim policy` del volumen persistente (persistent volume, PV) en el clúster de TKG original a `Retain` y, a continuación, elimine la PVC correspondiente.

Siga estos pasos para crear de forma estática una PVC en un nuevo clúster de TKG mediante la información del volumen subyacente de sobra.

### Procedimiento

1 Anote el nombre de la PVC original en el Supervisor.

Si vuelve a utilizar la PVC de un clúster de TKG antiguo, puede recuperar el nombre de la PVC de `volumeHandle` del objeto de PV anterior en el clúster de TKG.

2 Crear un PV.

En el archivo YAML, especifique los valores de los siguientes elementos:

- Para `storageClassName`, puede introducir el nombre de la clase de almacenamiento que utiliza su PVC en el Supervisor.
- Para `volumeHandle`, introduzca el nombre de PVC que obtuvo.

Si reutiliza un volumen de otro clúster de TKG, elimine los objetos de PVC y PV del clúster de TKG anterior antes de crear un PV en el nuevo clúster de TKG.

Utilice el siguiente manifiesto de YAML como ejemplo.

```
apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: static-tkg-block-pv
     annotations:
       pv.kubernetes.io/provisioned-by: csi.vsphere.vmware.com
   spec:
     storageClassName: gc-storage-profile
     capacity:
       storage: 2Gi
     accessModes:
       - ReadWriteOnce
     persistentVolumeReclaimPolicy: Delete
     claimRef:
       namespace: default
       name: static-tkg-block-pvc
     csi:
       driver: "csi.vsphere.vmware.com"
       volumeAttributes:
         type: "vSphere CNS Block Volume"
       volumeHandle: "supervisor-block-pvc-name"  # Enter the PVC name from the Supervisor.
```

3 Cree un PVC para que coincida con el objeto PV que creó.

Establezca la `storageClassName` en el mismo valor que en el PV.

```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: static-tkg-block-pvc
   spec:
     accessModes:
       - ReadWriteOnce
     resources:
```

```
      requests:
        storage: 2Gi
    storageClassName: gc-storage-profile
    volumeName: static-tkg-block-pv
```

4 Compruebe que la PVC esté enlazada al PV que creó.

```
$ kubectl get pv,pvc
 NAME                                  CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                        STORAGECLASS         REASON   AGE
 persistentvolume/static-tkg-block-pv    2Gi        RWO            Delete
Bound    default/static-tkg-block-pvc   gc-storage-profile           10s

 NAME                                        STATUS   VOLUME                CAPACITY
ACCESS MODES   STORAGECLASS         AGE
 persistentvolumeclaim/static-tkg-block-pvc   Bound    static-tkg-block-pv   2Gi
RWO            gc-storage-profile   10s
```

# Expansión de volúmenes persistentes para clústeres de Servicio TKG

Puede utilizar la función de expansión de volúmenes de Kubernetes para expandir un volumen de bloque persistente una vez creado. Los clústeres de Servicio TKG admiten la expansión de volúmenes en línea y sin conexión.

## Acerca de la expansión de volúmenes persistentes

De forma predeterminada, las clases de almacenamiento que aparecen en el entorno del clúster de TKG tienen `allowVolumeExpansion` establecido como `true`. Gracias a este parámetro, es posible modificar el tamaño de un volumen en línea y sin conexión.

Se considera que un volumen está sin conexión cuando no está asociado a un nodo o pod. Un volumen en línea es un volumen disponible en un nodo o pod.

El nivel de compatibilidad de la funcionalidad de expansión de volúmenes depende de la versión de vSphere. Puede expandir los volúmenes creados en las versiones anteriores de vSphere cuando actualice el entorno de vSphere a las versiones adecuadas que admitan las ampliaciones.

**Nota** Solo puede expandir volúmenes de bloques persistentes. Actualmente, vSphere IaaS control plane no admite la expansión de volúmenes para volúmenes ReadWriteMany.

Al expandir un volumen de bloque persistente, tenga en cuenta lo siguiente:

- Puede expandir los volúmenes hasta los límites especificados por las cuotas de almacenamiento. vSphere IaaS control plane admite solicitudes de cambio de tamaño consecutivas para un objeto de notificación de volumen persistente.
- Todos los tipos de almacenes de datos, incluidos VMFS, vSAN, vSAN Direct, vVols y NFS, admiten la expansión de volúmenes.
- Puede realizar una expansión de volúmenes para implementaciones o pods independientes.

- Puede cambiar el tamaño de los volúmenes aprovisionados estáticamente en un clúster de Tanzu Kubernetes Grid si los volúmenes tienen clases de almacenamiento asociadas.
- No puede expandir volúmenes creados como parte de StatefulSet.
- Si un disco virtual que crea una copia de seguridad de un volumen tiene instantáneas, no se puede cambiar su tamaño.
- vSphere IaaS control plane no admite la expansión de volúmenes para volúmenes en un árbol o migrados.

## Expandir un volumen persistente en modo en línea

Un volumen en línea es un volumen disponible en un nodo o pod. Como ingeniero de desarrollo y operaciones, puede expandir un volumen de bloque persistente en línea. Los clústeres de Tanzu Kubernetes Grid admiten la expansión de volúmenes en línea.

1 Busque la notificación de volumen persistente para cambiar su tamaño mediante el siguiente comando.

   Tenga en cuenta que, en este ejemplo, el tamaño del almacenamiento que utiliza el volumen es de 1 Gi.

```
$ kubectl get pv,pvc,pod
NAME                                                        CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   REASON   AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984   1Gi        RWO
Delete           Bound       default/block-pvc   block-sc                4m56s

NAME                              STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/block-pvc   Bound    pvc-5cd51b05-245a-4610-8af4-f07e77fdc984
1Gi        RWO            block-sc       5m3s

NAME            READY   STATUS    RESTARTS   AGE
pod/block-pod   1/1     Running   0          26s
```

2 Aplique una revisión a la PVC para aumentar su tamaño. Por ejemplo, aumente el tamaño a 2 Gi.

   Con esta acción se activa una expansión en el volumen asociado a la PVC.

```
$ kubectl patch pvc block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
persistentvolumeclaim/block-pvc edited
```

3 Compruebe que el tamaño de PVC y PV haya aumentado.

```
$ kubectl get pvc,pv,pod
NAME                              STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/block-pvc   Bound    pvc-5cd51b05-245a-4610-8af4-f07e77fdc984
2Gi        RWO            block-sc       6m18s
```

```
NAME                                                        CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   REASON   AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984   2Gi        RWO
Delete           Bound       default/block-pvc   block-sc                  6m11s

NAME             READY   STATUS    RESTARTS   AGE
pod/block-pod    1/1     Running   0          101s
```

4 Utilice vSphere Client para comprobar el nuevo tamaño del volumen persistente.

Consulte Supervisar el estado del volumen en un clúster de Tanzu Kubernetes Grid.

## Expandir un volumen persistente en modo sin conexión

Se considera que un volumen está sin conexión cuando no está asociado a un nodo o pod. Los clústeres de Tanzu Kubernetes Grid admiten la expansión de volúmenes sin conexión.

1 Cree una notificación de volumen persistente (PVC) para la clase de almacenamiento existente.

En el ejemplo, el tamaño del almacenamiento solicitado es 1 Gi.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-block-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: example-block-sc
```

```
kubectl apply -f example-block-pvc.yaml
```

2 Aplique una revisión a la PVC para aumentar su tamaño.

Si la PVC no está asociado a un nodo o no lo está usando un pod, utilice el siguiente comando para aplicar una revisión a la PVC. En este ejemplo, el aumento de almacenamiento solicitado es de 2 Gi.

Con esta acción se activa una expansión en el volumen asociado a la PVC.

```
kubectl patch pvc example-block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
```

3 Compruebe que el tamaño del volumen haya aumentado.

```
kubectl get pv
NAME                                       CAPACITY ACCESS MODES RECLAIM POLICY STATUS
CLAIM                       STORAGECLASS           REASON AGE
pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1   2Gi           RWO         Delete     Bound
default/example-block-pvc   example-block-sc              6m44s
```

4 Compruebe que el cambio de tamaño de la PVC esté pendiente.

**Nota** El tamaño de la PVC no cambia hasta que un pod utiliza la PVC.

El siguiente ejemplo muestra que el tamaño de la PVC no ha cambiado porque un pod no la ha utilizado. Si ejecuta `kubectl describe pvc`, verá la condición `FilesystemResizePending` aplicada en la PVC. Una vez que un pod la utilice, cambiará de tamaño.

```
kubectl get pvc
NAME                STATUS VOLUME                                     CAPACITY ACCESS
MODES   STORAGECLASS       AGE
example-block-pvc   Bound  pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1   1Gi
RWO       example-block-sc   6m57s
```

5 Cree un pod para utilizar la PVC.

Cuando el pod utiliza la PVC, se expande el sistema de archivos.

6 Compruebe que el tamaño de la PVC se haya modificado.

```
kubectl get pvc
NAME                STATUS VOLUME                                    CAPACITY ACCESS MODES
STORAGECLASS     AGE
example-block-pvc   Bound  pvc-24114458-9753-428e-9c90-9f568cb25788   2Gi         RWO
example-block-sc 2m12s
```

La condición `FilesystemResizePending` se ha eliminado de la PVC. La expansión del volumen se ha completado.

7 Utilice vSphere Client para comprobar el nuevo tamaño del volumen persistente.

Consulte Supervisar el estado del volumen en un clúster de Tanzu Kubernetes Grid.

# Administrar redes para clústeres de servicio TKG

17

En esta sección se proporciona información sobre cómo administrar redes para clústeres de servicio TKG.

Lea los siguientes temas a continuación:

- Instalar el servicio de proxy de administración de NSX
- Habilitar el adaptador Antrea-NSX para un clúster de servicio TKG
- Configurar la CNI predeterminada para los clústeres de Tanzu Kubernetes
- Personalizar la configuración del servicio TKG para clústeres de TKG
- Objetos de redes NSX para clústeres de TKG

## Instalar el servicio de proxy de administración de NSX

Puede utilizar el proxy de administración de NSX junto con el adaptador de Antrea-NSX para acceder a NSX Manager desde los clústeres de servicio de TKG basados en Antrea. El proxy de administración NSX es necesario cuando existe un aislamiento entre las redes de administración y carga de trabajo de Supervisor. El proxy se aplicará a todos los clústeres de TKGS creados después de implementar el servicio de proxy de administración de NSX.

### Caso de uso

El proxy de administración de NSX está pensado para su uso junto con el Habilitar el adaptador Antrea-NSX para un clúster de servicio TKG.

Si Supervisor se implementa con separación entre las redes de administración y carga de trabajo, como suele ser el caso, para llegar al plano de administración de NSX desde un clúster de TKGS configurado con el adaptador de Antrea-NSX, debe utilizar el proxy de administración de NSX. Cuando el sistema detecta este proxy, Supervisor lo pasa de forma automática a la configuración del adaptador de Antrea-NSX. Si el proxy no se encuentra instalado, el adaptador de Antrea-NSX no se podrá iniciar cuando se produzca el aislamiento de la red de administración.

Una vez implementado el servicio del proxy de administración de NSX, puede instalar el adaptador de Antrea-NSX. Consulte Habilitar el adaptador Antrea-NSX para un clúster de servicio TKG.

## Requisitos previos

El proxy de administración de NSX se instala como una instancia de servicio de supervisor. Para instalar el servicio del proxy, cumpla con los siguientes requisitos:

- vSphere 8 U3 (8.0.3) o versiones posteriores
- NSX 4.1 o versiones posteriores
- Supervisor está habilitado con redes NSX
- Privilegio Administrar servicios de supervisor en vCenter Server
- Estar familiarizado con Servicios de supervisor

## Descargar archivos YAML requeridos

Descargue los archivos YAML necesarios, incluidos los valores de datos y definición del servicio.

1. Vaya al sitio de distribución de servicios de supervisor en https://www.vmware.com/go/supervisor-service.
2. Desplácese hasta la sección Proxy de administración de NSX y descargue los siguientes archivos:
   a. El archivo de definición de servicio del proxy de administración de NSX: `nsx-management-proxy.yml`
   b. El archivo de configuración del servicio de proxy de administración de NSX: `nsx-management-proxy-data-values.yml`

## Registrar el proxy de administración de NSX como servicio

Siga estos pasos para registrar el proxy de administración de NSX como servicio de supervisor.

1. Con vSphere Client, desplácese hasta **Administración de carga de trabajo > Servicios**.
2. Seleccione **Agregar nuevo servicio > Agregar**.
3. Haga clic en **Cargar**.
4. Desplácese hasta el archivo `nsx-management-proxy.yml` que descargó y selecciónelo.
5. Compruebe que la definición del servicio proxy de administración de NSX se haya cargado correctamente.
6. Haga clic en **Finalizar**.
7. Compruebe que la tarjeta de registro del servicio de proxy de administración de NSX aparezca en la pestaña **Servicios**.

## Configurar el servicio de proxy de administración de NSX

Antes de instalar el servicio de proxy NSX Manager, actualice su archivo de valores de datos con los valores de configuración adecuados para su entorno.

1 En un editor de texto, abra el archivo `nsx-management-proxy-data-values.yml`.

2 Edite las propiedades para que coincidan con el entorno que se describe en la siguiente tabla.

3 Guarde los cambios.

| Nombre | Valor |
| --- | --- |
| *nsxManagers* | Lista de direcciones IP de NSX Manager (obligatoria).<br>Debe utilizar una dirección IP real y no la dirección IP virtual (VIP). Si utiliza un clúster de administración de NSX, debe incluir las 3 direcciones IP reales en la lista. |
| *loadBalancerIP* | IP del grupo de direcciones IP del equilibrador de carga de Supervisor (opcional).<br>La dirección IP de este campo se separa del CIDR de "Ingreso" para la red de cargas de trabajo. Cuando se crea una red de carga de trabajo, ya sea de forma inicial durante la habilitación de Supervisor, o más adelante al crear un espacio de nombres vSphere y anular la configuración de red, hay una opción de configuración para "Ingreso" que acepta un bloque CIDR de direcciones IP que se utiliza a fin de asignar direcciones IP para los servicios que se publican a través del ingreso y el equilibrador de carga del tipo de servicio en todos los espacios de nombres de vSphere creados en esa instancia de Supervisor.<br>Si no se especifica el campo "loadBalancerIP", el sistema asignará de forma automática una dirección IP disponible del rango de CIDR de "Ingreso".<br>Si se especifica "loadBalancerIP", debe estar dentro del rango de CIDR de "Ingreso" y no debe entrar en conflicto con las direcciones IP que ya se encuentran asignadas.<br>Puede ver las direcciones IP de "Ingreso" asignadas mediante el comando "kubectl get services -o wide -A" en el espacio de nombres de vSphere. Las direcciones IP se encuentran en la columna "EXTERNAL-IP" de los resultados de kubectl. |

## Instalar el servicio de proxy de administración de NSX

Siga estos pasos para instalar el servicio de proxy de administración de NSX.

1 Desplácese hasta la pantalla **Administración de carga de trabajo > > Servicios**.

2 En la tarjeta de servicio **proxy de administración de NSX**, seleccione **Acciones > Instalar en supervisor**.

3 Seleccione la pestaña **Disponible**.

4 Copie y pegue el contenido del archivo `nsx-management-proxy-data-values.yml` que editó en el campo de entrada "Configuración del servicio YAML".

5 Haga clic en **Aceptar** para continuar con la instalación de Harbor.

6 Supervise y compruebe la instalación.

Para supervisar la instalación, compruebe el campo Supervisores en la tarjeta de servicio del proxy de administración de NSX. Debería ver el número junto a los incrementos de Supervisores. El estado del servicio será Configurando hasta que se alcance el estado deseado. Cuando se alcance el estado deseado, el estado del servicio cambiará a Configurado.

7 Compruebe que haya un espacio de nombres de vSphere para el proxy de administración de NSX.

Una vez que esté instalado el proxy de administración de NSX, se crea un espacio de nombres de vSphere para la instancia de servicio.

8 Obtenga la dirección IP del equilibrador de carga del proxy.

Puede ver la dirección IP del equilibrador de carga del proxy en la pestaña **Red** del espacio de nombres de vSphere del proxy de administración de NSX.

## Solucionar problemas del servicio de proxy de administración de NSX

Si recibe el siguiente error, significa que el entorno no es compatible. Asegúrese de que el entorno cumpla con las versiones enumeradas en la sección de requisitos previos.

```
Creation of Supervisor Service with ID nsx-management-proxy.nsx.vmware.com is not allowed.
Only service IDs defined in the allow-list file /etc/vmware/wcp/supervisor-services-allow-
list.txt are allowed.
```

# Habilitar el adaptador Antrea-NSX para un clúster de servicio TKG

Consulte este tema para habilitar el adaptador Antrea-NSX, que le permite integrar un clúster de servicio TKG que utiliza la CNI de Antrea con NSX Manager para la supervisión y la administración de redes.

## Requisitos previos para el adaptador Antrea-NSX

Cumpla los siguientes requisitos previos:

- vSphere 8 U3 (8.0.3) o versiones posteriores
- NSX 4.1 o versiones posteriores
- Supervisor está habilitado con redes NSX
- Servicio TKG 3.0 o versiones posteriores

- versión de Tanzu Kubernetes 1.28.x para vSphere 8.x o versiones posteriores
- Instalar el servicio de proxy de administración de NSX si hay aislamiento entre las redes de administración y de cargas de trabajo, que es la topología típica de Supervisor

## Requisitos para el adaptador Antrea-NSX

El adaptador Antrea-NSX permite integrar un clúster de servicio TKG basado en Antrea con NSX Manager. Una vez configurado el adaptador, puede utilizar NSX Manager para administrar el comportamiento de redes del clúster. Para obtener más información sobre las capacidades de la integración Antrea-NSX, consulte Integración de clústeres de Kubernetes con CNI de Antrea en la Guía de administración de NSX 4.1.

Debe habilitar el adaptador Antrea-NSX para cada clúster de servicio TKG que desee integrar con NSX. En otras palabras, existe una relación de 1 a 1 entre un adaptador y un clúster. Además, solo puede utilizar el adaptador Antrea-NSX con nuevas implementaciones de clústeres. Debe habilitar el adaptador antes de crear el clúster de servicio TKG y debe incluir el nombre del clúster que se va a crear en la definición de recursos del adaptador.

Si las puertas de enlace de nivel 0 o de nivel 1 de NSX para clústeres de TKGS están configuradas con una IP de SNAT, todas las conexiones de NSX Antrea compartirán una única IP de origen. NSX interpretará esta cantidad de conexiones del plano de control desde la misma IP y quitará las conexiones. Si este es el caso, debe cambiar manualmente las reglas de firewall de UA NSX. Consulte los siguientes artículos de la base de conocimientos para obtener detalles: https://knowledge.broadcom.com/external/article?articleNumber=317179.

## Habilitar el adaptador Antrea-NSX

Siga estos pasos para habilitar el adaptador Antrea-NSX.

1 Autentíquese con Supervisor mediante kubectl.

```
kubectl vsphere login --server=SUPERVISOR-CONTROL-PLANE-IP-ADDRESS-or-FQDN --vsphere-
username USERNAME
```

2 Cambie el contexto a la instancia de espacio de nombres de vSphere de destino donde aprovisionará el clúster de servicio TKG.

```
kubectl config use-context vsphere-namespace
```

3 Cree la definición de recurso personalizado `AntreaConfig.yaml`.

```
#AntreaConfig.yaml
apiVersion: cni.tanzu.vmware.com/v1alpha1
kind: AntreaConfig
metadata:
 name: tkgs-cluster-name-antrea-package #prefix required
```

```
 namespace: tkgs-cluster-ns
spec:
  antreaNSX:
    enable: true #false by default
```

Donde:

- El elemento `antreaNSX.enable.true` habilita el adaptador Antrea-NSX. De forma predeterminada, este campo es falso.
- El elemento `metadata.name` incluye el sufijo obligatorio `-antrea-package` y viene precedido por el nombre exacto del clúster de servicio TKG que creará y que utilizará el adaptador.

4 Aplique la definición de recurso personalizado AntreaConfig.

```
kubectl apply -f AntreaConfig.yaml
```

5 Aprovisione el clúster de servicio TKG.

El adaptador se puede utilizar con la API v1alpha3 o la API v1beta1.

Consulte Flujo de trabajo para aprovisionar clústeres de TKG mediante Kubectl.

6 Inicie sesión en NSX Manager para comprobar que el adaptador funcione.

Consulte la documentación de NSX 4.1 para obtener más instrucciones.

# Configurar la CNI predeterminada para los clústeres de Tanzu Kubernetes

La interfaz de red de contenedor (Container Network Interface, CNI) predeterminada para los clústeres de Tanzu Kubernetes es Antrea. Con vSphere Client, puede cambiar la CNI predeterminada.

## CNI predeterminada

vSphere IaaS control plane admite dos opciones de la CNI para los clústeres de TKG: Antrea y Calico. La CNI predeterminada definida por el sistema es Antrea.

Puede cambiar la CNI predeterminada mediante vSphere Client. Para establecer la CNI predeterminada, complete el siguiente procedimiento.

---

**Precaución** Cambiar la CNI predeterminada es una operación global. El valor predeterminado recién establecido se aplica a todos los clústeres de TKG nuevos que cree el servicio. Los clústeres existentes no se modifican.

---

1 Inicie sesión en su entorno de vSphere IaaS control plane mediante vSphere Client.

2 Seleccione **Administración de cargas de trabajo** y **Supervisores**.

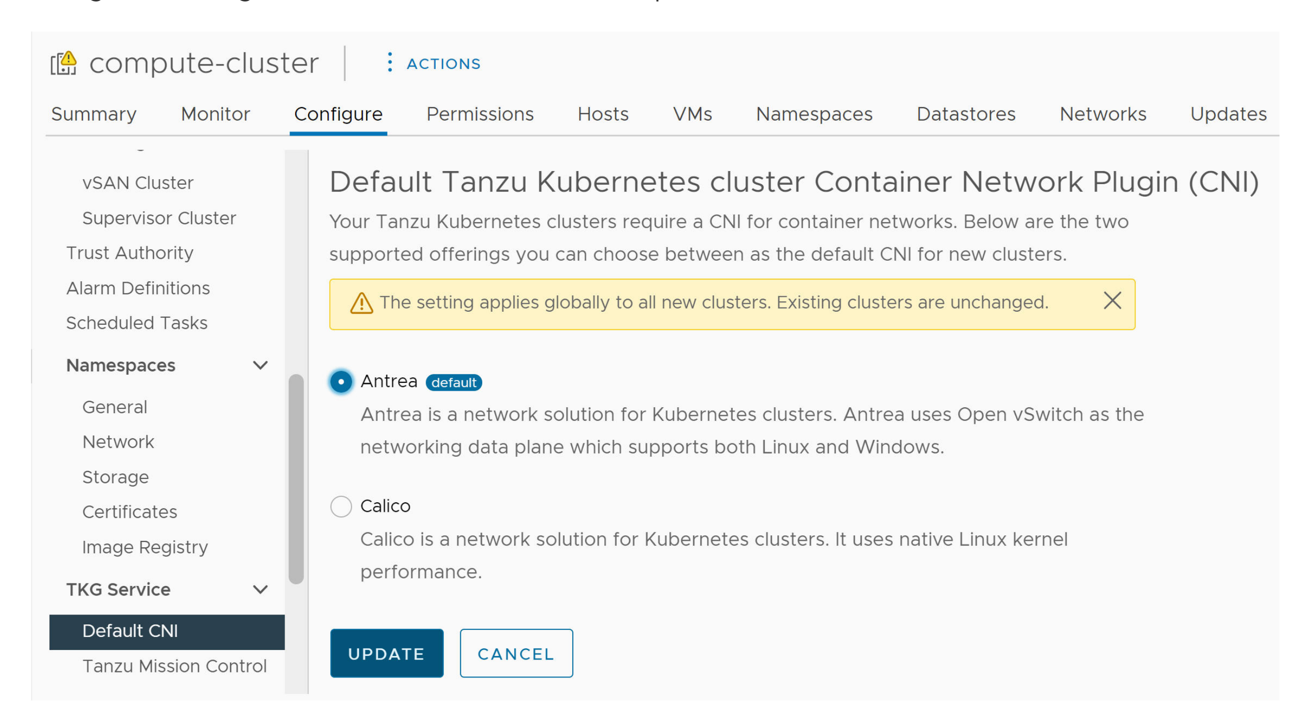
3 Seleccione la instancia de Supervisor de la lista.

4 Seleccione **Configurar** y **Servicio de TKG > CNI predeterminada**

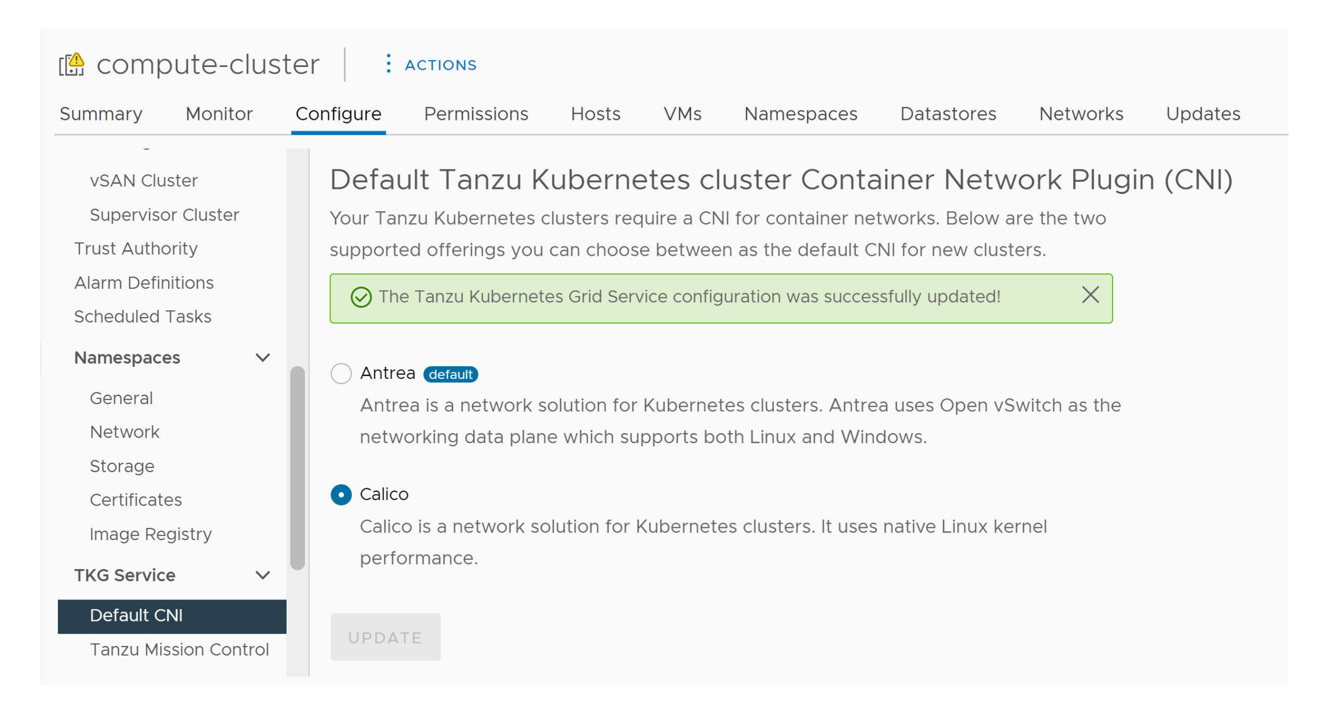
5 Elija la CNI predeterminada para los nuevos clústeres.

6 Haga clic en **Actualizar**.

La siguiente imagen muestra la selección de CNI predeterminada.




La siguiente imagen muestra cómo cambiar la selección de CNI de Antrea a Calico.



# Personalizar la configuración del servicio TKG para clústeres de TKG

Puede personalizar la Configuración del servicio TKG para clústeres de TKG de aprovisionados mediante la API v1alpha3, incluida la interfaz de red de contenedor (Container Network Interface, CNI), el servidor proxy y los certificados TLS.

## Acerca de la personalización de TkgServiceConfiguration

Si desea configurar los ajustes globales para clústeres de Tanzu Kubernetes, modifique TkgServiceConfiguration. Esta configuración permite establecer la CNI predeterminada, agregar un servidor proxy global y agregar uno o varios certificados TLS de confianza.

**Precaución** La personalización de TkgServiceConfiguration es una operación global. Cualquier cambio que realice en el objeto `TkgServiceConfiguration` se aplicará a todos los clústeres de TKG aprovisionados por ese servicio. Si se inicia una actualización gradual, ya sea de forma manual o mediante actualización, los clústeres se actualizan según la especificación de servicio modificada.

**Nota** La personalización de TkgServiceConfiguration se aplica a los clústeres de Tanzu Kubernetes aprovisionados con la API v1alpha3. No se aplica a los clústeres aprovisionados con la API v1beta1.

## Especificación TkgServiceConfiguration

La especificación `TkgServiceConfiguration` proporciona campos para configurar la instancia de Tanzu Kubernetes Grid.

**Importante** Un nombre de clave válido solo debe constar de caracteres alfanuméricos, un guion (como `key-name`), un guion bajo (como `KEY_NAME`) o un punto (como `key.name`). No puede utilizar un espacio en un nombre de clave.

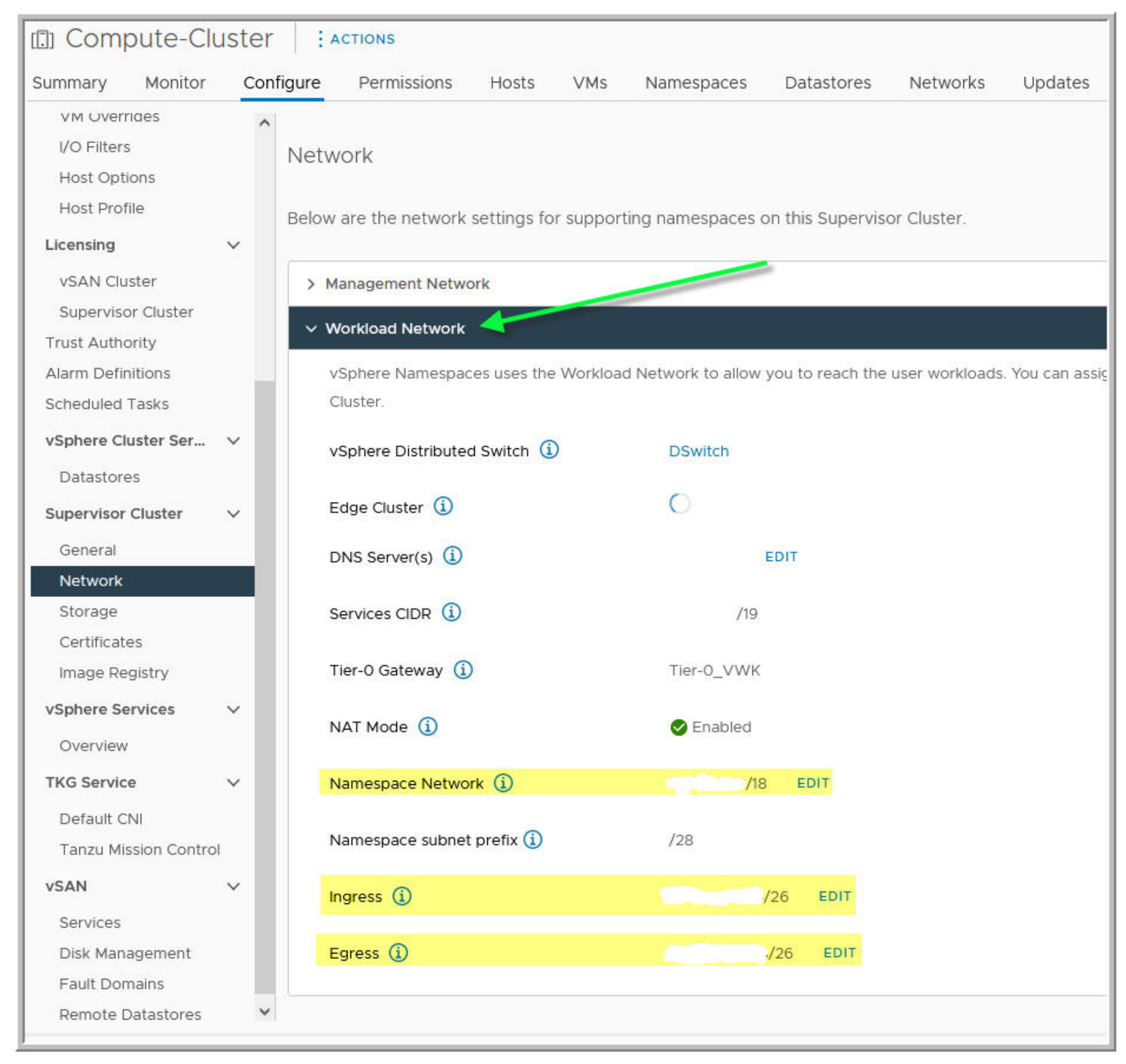
```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  defaultCNI: string
  proxy:
    httpProxy: string
    httpsProxy: string
    noProxy: [string]
  trust:
    additionalTrustedCAs:
      - name: string
        data: string
  defaultNodeDrainTimeout: time
```

## Especificación de TkgServiceConfiguration anotada

El siguiente YAML enumera y describe los campos configurables para cada uno de los parámetros de especificación de `TkgServiceConfiguration`.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TkgServiceConfiguration
#valid config key must consist of alphanumeric characters, '-', '_' or '.'
metadata:
  name: tkg-service-configuration-spec
spec:
  #defaultCNI is the default CNI for all Tanzu Kubernetes
  #clusters to use unless overridden on a per-cluster basis
  #supported values are antrea, calico, antrea-nsx-routed
  #defaults to antrea
  defaultCNI: string
  #proxy configures a proxy server to be used inside all
  #clusters provisioned by this TKGS instance
  #if implemented all fields are required
  #if omitted no proxy is configured
  proxy:
    #httpProxy is the proxy URI for HTTP connections
    #to endpionts outside the clusters
    #takes the form http://<user>:<pwd>@<ip>:<port>
    httpProxy: string
    #httpsProxy is the proxy URI for HTTPS connections
    #to endpoints outside the clusters
    #takes the frorm http://<user>:<pwd>@<ip>:<port>
    httpsProxy: string
    #noProxy is the list of destination domain names, domains,
    #IP addresses, and other network CIDRs to exclude from proxying
    #must include from Workload Network: [Namespace Netowrk, Ingress, Egress]
    noProxy: [string]
  #trust configures additional trusted certificates
  #for the clusters provisioned by this TKGS instance
  #if omitted no additional certificate is configured
  trust:
    #additionalTrustedCAs are additional trusted certificates
    #can be additional CAs or end certificates
    additionalTrustedCAs:
      #name is the name of the additional trusted certificate
      #must match the name used in the filename
      - name: string
        #data holds the contents of the additional trusted cert
        #PEM Public Certificate data encoded as a base64 string
        data: string
  #defaultNodeDrainTimeout is the total amount of time the
  #controller spends draining a node; default is undefined
  #which is the value of 0, meaning the node is drained
  #without any time limitations; note that `nodeDrainTimeout`
  #is different from `kubectl drain --timeout`
  defaultNodeDrainTimeout: time
```

Figura 17-1. Valores de noProxy necesarios




**Nota** Si se configura un proxy global en `TkgServiceConfiguration`, esa información de proxy se propaga al manifiesto del clúster después de la implementación inicial del clúster. La configuración global del proxy se agrega al manifiesto del clúster solo si no hay ningún campo de configuración de proxy presente cuando se crea el clúster. En otras palabras, la configuración por clúster tiene prioridad y sobrescribirá la configuración global del proxy.

## Especificación de TkgServiceConfiguration de ejemplo

El siguiente YAML enumera y describe los campos configurables para cada uno de los parámetros de especificación de `TkgServiceConfiguration`.

```
apiVersion: run.tanzu.vmware.com/v1alpha3
kind: TkgServiceConfiguration
```

```
metadata:
  name: tkgserviceconfiguration_example
spec:
  defaultCNI: calico
  proxy:
    #supported format is `http://<user>:<pwd>@<ip>:<port>`
    httpProxy: http://admin:PaSsWoRd@10.66.100.22:80
    httpsProxy: http://admin:PaSsWoRd@10.66.100.22:80
    #noProxy vaulues are from Workload Network: [Namespace Network, Ingress, Egress]
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      #name is the name of the public cert
      - name: CompanyInternalCA-1
      #data is base64-encoded string of a PEM encoded public cert
        data: LS0tLS1C...LS0tCg==
        #where "..." is the middle section of the long base64 string
      - name: CompanyInternalCA-2
        data: MTLtMT1C...MT0tPg==
  defaultNodeDrainTimeout: 0
```

## Cómo editar TkgServiceConfiguration

Consulte el siguiente procedimiento para editar la especificación de TkgServiceConfiguration.

1 Configure la edición de Kubectl. Consulte Configurar un editor de texto para Kubectl.

2 Realice la autenticación con Supervisor.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

3 Cambie el contexto al espacio de nombres de vSphere de destino.

```
kubectl config use-context vSphere-Namespace
```

4 Obtenga la especificación de TkgServiceConfiguration.

```
kubectl get tkgserviceconfigurations
```

5 Cargue para editar la especificación de TkgServiceConfiguration.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

El sistema abre la especificación `tkg-service-configuration` en el editor de texto predeterminado que definen las variables de entorno KUBE_EDITOR o EDITOR.

6 Edite TkgServiceConfiguration según sus requisitos.

7 Para aplicar los cambios, guarde el archivo en el editor de texto. Para cancelar, cierre el editor sin guardar.

Al guardar el cambio en el editor de texto, kubectl actualiza la especificación de servicio de `tkg-service-configuration`.

8 Compruebe que la especificación de TkgServiceConfiguration se haya actualizado.

```
kubectl get tkgserviceconfigurations
```

## Propagar cambios de configuración global a clústeres existentes

Los ajustes que se realizaron a nivel global en `TkgServiceConfiguration` no se propagan automáticamente a los clústeres existentes. Por ejemplo, si realiza cambios en los ajustes de `proxy` o `trust` en `TkgServiceConfiguration`, los cambios no afectarán los clústeres que ya están aprovisionados.

Para propagar manualmente un cambio global a un clúster existente, debe aplicar una revisión al clúster de Tanzu Kubernetes para que herede los cambios realizados en `TkgServiceConfiguration`.

Por ejemplo:

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"proxy\": null}}}}"
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"trust\": null}}}}"
```

# Objetos de redes NSX para clústeres de TKG

En este tema se enumeran los objetos de red creados para un clúster de TKG cuando se utiliza Supervisor con redes NSX.

## Objetos de redes NSX para clústeres de TKG

Cada clúster de TKG debe tener los siguientes recursos de red: una red virtual, una interfaz de red virtual y un servicio de máquina virtual.

El sistema aprovisiona automáticamente un equilibrador de carga NSX integrado cuando está habilitado vSphere IaaS control plane y se implementa una instancia de Supervisor. Este equilibrador de carga es para el plano de control de Supervisor y proporciona acceso al servidor de API de Kubernetes.

Cuando se crea un servicio de Kubernetes de tipo LoadBalancer para un clúster de TKG, se aprovisiona un equilibrador de carga integrado NSX para ese servicio.

| Objeto de red | Recursos de red | Descripción |
|---|---|---|
| Red virtual | Enrutador de nivel 1 y segmento vinculado | Red de nodo para el clúster |
| VirtualNetworkInterface | Puerto lógico en el segmento | Interfaz de red de nodo para nodos del clúster |
| Servicio de máquina virtual | N/C | Se crea el servicio de máquina virtual y se traduce a un servicio k8s. |

| Objeto de red | Recursos de red | Descripción |
| --- | --- | --- |
| Servicio | Servidor de equilibrador de carga con instancia de servidor virtual y grupo de servidores asociado (grupo de miembros) | Se crea un servicio de Kubernetes de tipo equilibrador de carga para acceder al servidor de API del clúster de TKG. |
| Endpoints | Los miembros del endpoint (nodos de plano de control del clúster de TKG) deben estar en el grupo de miembros. | Se crea un endpoint para incluir todos los nodos del plano de control del clúster de TKG. |
| Servicio de máquina virtual en supervisor | N/C | Se crea un servicio de máquina virtual en supervisor y se traduce en un servicio de Kubernetes en supervisor |
| Servicio de equilibrador de carga en supervisor | Servidor virtual en el equilibrador de carga del clúster de TKG y un grupo de miembros asociado. | Se crea el servicio de equilibrador de carga en supervisor para acceder a este tipo de servicio de equilibrador de carga |
| Endpoints en Supervisor | Los miembros del endpoint (nodos de trabajo del clúster de TKG) deben estar en el grupo de miembros en NSX. | Se crea un endpoint para incluir todos los nodos de trabajo del clúster de TKG |
| Servicio de equilibrador de carga en el clúster de TKG | N/C | El servicio del equilibrador de carga en el clúster de TKG implementado por el usuario debe tener su estado actualizado con la IP del equilibrador de carga |

## Redes de nodos

Cada clúster de TKG debe tener creados los siguientes objetos de red y recursos de NSX asociados.

| Objeto de red | Recursos de NSX | Descripción | IPAM |
| --- | --- | --- | --- |
| Red virtual | Puerta de enlace de nivel 1 y segmento vinculado | Red de nodo para el clúster de TKG | Se ha asignado la dirección IP de SNAT |
| VirtualNetworkInterface | Puerto lógico en el segmento vinculado | Interfaz de red de nodo para nodos de clúster de TKG | A cada nodo se le asigna una dirección IP |

## Equilibrador de carga del plano de control

| Objeto de red | Recursos de red | Descripción | IPAM |
|---|---|---|---|
| Servicio de máquina virtual | N/C | VirtualMachineService se crea y se traduce en un servicio de Kubernetes. | Incluye la VIP del equilibrador de carga. |
| Servicio | Servidor de equilibrador de carga con instancia de servidor virtual y grupo de servidores asociado (grupo de miembros) | Se crea un servicio de Kubernetes de tipo equilibrador de carga para acceder al servidor de API del clúster de TKG. | Se ha asignado una dirección IP externa. |
| Endpoints | Los miembros del endpoint son los nodos del plano de control del clúster de TKG y deben estar en el grupo de miembros. | Se crea un endpoint para incluir todos los nodos del plano de control del clúster de TKG. | N/C |

## Equilibradores de carga de NSX

Para cada clúster de TKG creado, el sistema crea una única instancia de un equilibrador de carga de NSX pequeño. Este equilibrador de carga contiene los objetos enumerados en la siguiente tabla:

| Número de objeto | Descripción |
|---|---|
| 1 | Servidor virtual (VS, Virtual Server) para acceder a la API del plano de control de Kubernetes en el puerto 8443. |
| 1 | Grupo de servidores que contiene los 3 nodos del plano de control de Kubernetes. |
| 1 | VS para el controlador de entrada HTTP. |
| 1 | VS para el controlador de entrada HTTPS. |

## Reglas NAT

Para cada clúster de TKG creado, el sistema define las siguientes reglas NAT de NSX en el enrutador lógico de nivel 0:

| Número de objeto | Descripción |
|---|---|
| 1 | Regla SNAT creada para cada espacio de nombres de Kubernetes mediante 1 IP del grupo de direcciones IP flotantes como dirección IP traducida. |
| 1 | (Solo topología NAT) Regla SNAT creada para cada clúster de Kubernetes mediante 1 IP del grupo de direcciones IP flotantes como dirección IP traducida. La subred del clúster de Kubernetes se deriva del bloque de direcciones IP de los nodos mediante una máscara de red /24. |

## Reglas DFW

Para cada clúster de TKG creado, el sistema define las siguientes reglas de firewall distribuido de NSX:

| Número de objeto | Descripción |
|---|---|
| 1 | Regla DFW para `kube-dns`, aplicada al puerto lógico del pod de CoreDNS: |
| 1 | Regla DFW para el validador en el espacio de nombres, aplicada al puerto lógico del pod del validador: |

# Administrar la seguridad para clústeres de servicio TKG

18

En esta sección se proporciona información sobre cómo administrar la seguridad para clústeres de servicio TKG.

Lea los siguientes temas a continuación:



## Seguridad para clústeres de servicio TKG

El servicio TKG con Supervisor aprovecha las funciones de seguridad de vSphere y aprovisiona los clústeres de carga de trabajo que son seguros de forma predeterminada.

vSphere IaaS control plane es un módulo complementario para vSphere que puede aprovechar las funciones de seguridad integradas en vCenter Server y ESXi. Para obtener más información, consulte la documentación de Seguridad de vSphere.

Supervisor cifra todos los secretos almacenados en la base de datos (etcd). Los secretos se cifran a través de un archivo de clave de cifrado local, que vCenter Server proporciona en el arranque. La clave de descifrado se almacena en la memoria (tempfs) en los nodos del plano de control de Supervisor y en el disco de forma cifrada dentro de la base de datos de vCenter Server. La clave está disponible en texto no cifrado para los usuarios raíz de cada sistema.

El mismo modelo de cifrado se aplica a los datos de la base de datos (etcd) que está instalada en cada plano de control de clúster de TKG. Todas las conexiones etcd se autentican con certificados que se generan en la instalación y se rotan durante las actualizaciones. Actualmente no es posible rotar o actualizar manualmente los certificados. Los secretos que se encuentran en la base de datos de cada clúster de carga de trabajo se almacenan en texto no cifrado.

Un clúster de TKG no tiene credenciales de infraestructura. Las credenciales que se almacenan en un clúster de TKG solo son suficientes para acceder al espacio de nombres de vSphere donde el clúster de TKG es tenant. Por ello, no existe la posibilidad de realizar la escalación de privilegios para los operadores de clústeres ni los usuarios.

El token de autenticación utilizado para acceder a un clúster de TKG tiene un ámbito tal que el token no se puede usar para acceder al Supervisor o a otros clústeres de TKG. De este modo, se evita que los operadores del clúster, o los individuos que puedan intentar poner en peligro un clúster, utilicen el acceso de nivel raíz para capturar un token de administrador de vSphere cuando inicien sesión en un clúster de TKG.

Un clúster de TKG está protegido de forma predeterminada. A partir de versión de Tanzu Kubernetes v1.25, los clústeres de TKG tienen la controladora de administración de seguridad de pods (Pods Security Adminission, PSA) habilitada de forma predeterminada. Para versiones de Tanzu Kubernetes hasta la versión 1.24, la directiva de seguridad de pods (Pod Security Policy, PSP) restrictiva está disponible para todo clúster de TKG. Si los desarrolladores necesitan ejecutar contenedores raíz o pods con privilegios, al menos un administrador de clústeres deberá crear un objeto RoleBinding que otorgue acceso de usuario a la PSP con privilegios predeterminada.

# Configurar PSA para TKR 1.25 y versiones posteriores

Las versiones 1.25 y posteriores de Tanzu Kubernetes admiten la controladora de admisión de seguridad de pods (Pod Security Admission, PSA). Con PSA, puede aplicar de manera uniforme la seguridad de pods mediante etiquetas de espacio de nombres.

## PSA habilitada en TKR 1.25 y versiones posteriores

La controladora de admisión de seguridad de pods es una controladora de Kubernetes que permite aplicar estándares de seguridad a los pods que se ejecutan en los clústeres de TKG. De forma predeterminada, las Tanzu Kubernetes versiones 1.25 y posteriores habilitan la controladora de PSA. La controladora de PSA reemplaza a la controladora de la directiva de seguridad de pods (Pod Security Policy, PSP), que está obsoleta y se eliminó. Consulte también Configurar PSP para TKR 1.24 y versiones anteriores.

Tanzu Kubernetes v1.25 es una versión de transición con PSA configurada como advertencia. A partir de Tanzu Kubernetes v1.26, PSA entrará en vigor. Debe planificar la migración de las cargas de trabajo de pods de PSP a PSA antes de actualizar los clústeres de TKG. Para obtener instrucciones, consulte Migrar de la directiva de seguridad de pods a la controladora de admisión de seguridad de pods integrada.

## Configurar PSA en todo el clúster

A partir de vSphere 8 Update 3, puede configurar PSA en todo el clúster mediante la variable ClusterClass de `podSecurityStandard` que está disponible con la API v1beta1. Consulte API de clúster v1beta1 .

## Modos de PSA

La controladora de PSA admite tres modos de seguridad de pods: `enforce`, `audit` y `warn`. En la tabla se enumera y describe cada modo de PSA.

**Tabla 18-1. Modos de PSA**

| MODO | Descripción |
| --- | --- |
| `enforce` | Las infracciones de seguridad hacen que se rechace el pod. |
| `audit` | Las infracciones de seguridad generan la adición de una anotación de auditoría al evento consignado en el registro de auditoría, pero por lo demás están permitidas. |
| `warn` | Las infracciones de seguridad activan una advertencia orientada al usuario, pero por lo demás están permitidas. |

## Estándares de PSA

La controladora de PSA define tres niveles de estándares de seguridad de pods destinados a cubrir el espectro de seguridad. Los estándares de seguridad de pods son acumulativos y van de permisivos a restrictivos. En la tabla se enumera y describe cada estándar de PSA.

**Tabla 18-2. Estándares de PSA**

| NIVEL | Descripción |
| --- | --- |
| `privileged` | Control sin restricciones, que brinda el nivel más amplio posible de permisos. Este estándar de seguridad permite escalaciones de privilegios conocidos. |
| `baseline` | Control poco restrictivo que impide escalaciones de privilegios conocidos. Este estándar de seguridad permite la configuración predeterminada (mínimamente especificada) de pods. |
| `restricted` | Control muy restringido, que sigue las prácticas recomendadas de protección de pods. |

## Etiquetas de espacio de nombres de PSA

La controladora de PSA aplica la seguridad de pods en el nivel del espacio de nombres de Kubernetes. Las etiquetas de espacio de nombres se utilizan para definir los modos y los niveles de PSA que desea utilizar para los pods en un espacio de nombres determinado.

Kubernetes proporciona un conjunto de etiquetas que pueden utilizarse para definir cuáles de los estándares se utilizarán para un espacio de nombres. La etiqueta que aplique define la acción que realiza el plano de control de Kubernetes si se detecta una posible infracción de la PSA. Para un espacio de nombres de Kubernetes dado, puede configurar algún modo o todos, o incluso establecer un nivel diferente para los distintos modos.

La sintaxis de las etiquetas del espacio de nombres de PSA es la siguiente:

```
# MODE must be one of `enforce`, `audit`, or `warn`.
# LEVEL must be one of `privileged`, `baseline`, or `restricted`.
pod-security.kubernetes.io/<MODE>=<LEVEL>
```

También puede aplicar una etiqueta de versión por modo que se pueda utilizar para fijar el estándar de seguridad a la versión de Kubernetes. Consulte Aplicar estándares de seguridad de pods con etiquetas de espacio de nombres para obtener más información.

## PSA predeterminada para clústeres de TKG

De forma predeterminada, los clústeres de TKG aprovisionados con Tanzu Kubernetes versión 1.25 tienen modos PSA `warn` y `audit` establecidos en `restricted` para espacios de nombres que no son del sistema. Esta es una configuración sin fuerza: la controladora de PSA genera una advertencia y una notificación de auditoría si un pod infringe la seguridad, pero no se rechaza el pod.

De forma predeterminada, los clústeres de TKG aprovisionados con Tanzu Kubernetes versión 1.26 y posteriores tienen el modo PSA `enforce` establecido en `restricted` para espacios de nombres que no son del sistema. Si un pod infringe la seguridad, se rechaza. Debe configurar PSA en el espacio de nombres para ejecutar pods con un control menos restrictivo.

**Importante** Algunos pods del sistema que se ejecutan en los espacios de nombres kube-system, tkg-system y vmware-system-cloud-provider requieren privilegios elevados. Estos espacios de nombres se excluyen de la seguridad de pods. Además, no se puede cambiar la seguridad de los pods en los espacios de nombres del sistema.

En la tabla se muestra la configuración de PSA predeterminada para clústeres de TKG.

**Tabla 18-3. PSA predeterminada para clústeres de TKG**

| Versión de TKr | PSA predeterminada |
| --- | --- |
| TKr v1.25 | modo: advertencia \| nivel: restringido<br>modo: auditoría \| nivel: restringido<br>modo: aplicación \| nivel: no establecido |
| TKr v1.26 y versiones posteriores | modo: aplicación \| nivel: restringido |

## Configurar PSA mediante etiquetas de espacio de nombres

Para Tanzu Kubernetes versión 1.25, utilice el siguiente comando de ejemplo a fin de cambiar los niveles de seguridad de un espacio de nombres determinado, de modo que no se generen las advertencias de PSA ni las notificaciones de auditoría.

```
kubectl label --overwrite ns NAMESPACE pod-security.kubernetes.io/audit=privileged
kubectl label --overwrite ns NAMESPACE pod-security.kubernetes.io/warn=privileged
```

Para Tanzu Kubernetes versiones 1.26 y posteriores, utilice el siguiente comando de ejemplo para cambiar el estándar PSA de `restricted` a `baseline`.

```
kubectl label --overwrite ns NAMESPACE pod-security.kubernetes.io/enforce=baseline
```

Por ejemplo, para aplicar el estándar `baseline` en el espacio de nombres predeterminado:

```
kubectl label --overwrite ns default pod-security.kubernetes.io/enforce=baseline
```

Para Tanzu Kubernetes versiones 1.26 y posteriores, utilice el siguiente comando de ejemplo para cambiar el estándar PSA de `restricted` a `privileged`.

```
kubectl label --overwrite ns NAMESPACE pod-security.kubernetes.io/enforce=privileged
```

Por ejemplo, para aplicar el estándar `privileged` en el espacio de nombres predeterminado:

```
kubectl label --overwrite ns default pod-security.kubernetes.io/enforce=privileged
```

Para Tanzu Kubernetes versiones 1.26 y posteriores, utilice los siguientes comandos de ejemplo para relajar la PSA en todos los espacios de nombres que no sean del sistema.

```
kubectl label --overwrite ns --all pod-security.kubernetes.io/enforce=privileged
```

```
kubectl label --overwrite ns --all pod-security.kubernetes.io/warn=restricted
```

## Configurar el contexto de seguridad para pods individuales

Si intenta ejecutar un pod que infringe la PSA, se reactiva un mensaje de error que lo indica. Por ejemplo:

```
{"opType":"CREATE_POD","succeeded":false,"err":"creating pod example-pod: pods
\"example-pod\" is forbidden: violates PodSecurity \"restricted:latest\":
allowPrivilegeEscalation != false (container \"example-container\" must set
securityContext.allowPrivilegeEscalation=false), unrestricted capabilities
(container \"example-container\" must set securityContext.capabilities.drop=[\"ALL\"]),
runAsNonRoot != true (pod or container \"example-container\" must set
securityContext.runAsNonRoot=true),
seccompProfile (pod or container \"example-container\" must set
securityContext.seccompProfile.type to
\"RuntimeDefault\" or \"Localhost\")","events":[]}
```

En lugar de establecer PSA en un espacio de nombres completo, puede configurar el contexto de seguridad para un pod individual. Para permitir que se ejecute un pod individual, en la especificación del pod, puede establecer el contexto de seguridad de la siguiente manera:

```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
```

```
  - image: gcr.io/google_containers/busybox:1.24
    name: example-container
    command: ["/bin/sh", "-c", "echo 'hello' > /mnt/volume1/index.html && chmod
o+rX /mnt /mnt/volume1/index.html && while true ; do sleep 2 ; done"]
    securityContext:
      allowPrivilegeEscalation: false
      runAsNonRoot: true
      seccompProfile:
        type: "RuntimeDefault"
      capabilities:
        drop: [all]
    volumeMounts:
    - name: example-volume-mount
      mountPath: /mnt/volume1
  restartPolicy: Never
  volumes:
  - name: example-volume
    persistentVolumeClaim:
    claimName: example-pvc
```

# Configurar PSP para TKR 1.24 y versiones anteriores

TKG en Supervisor admite la seguridad de pods mediante la controladora de admisión de la directiva de seguridad de pods, que está habilitada de forma predeterminada para los clústeres de Tanzu Kubernetes que utilizan TKR v1.24 y versiones anteriores.

## Requisito previo: TKR 1.24 y versiones anteriores

El contenido de este tema se aplica a los clústeres de TKG en Supervisor que se aprovisionan con TKR v1.24 y versiones anteriores. De forma predeterminada, versiones de Tanzu Kubernetes habilita la controladora de admisión de la directiva de seguridad de pods.

La controladora de admisión de la directiva de seguridad de pods es la controladora de admisión de seguridad de pods. Los clústeres de TKG en Supervisor aprovisionados con TKR v1.25 y versiones posteriores habilitan la controladora de admisión de seguridad de pods. Consulte Configurar PSA para TKR 1.25 y versiones posteriores.

## Controladora de admisión de la directiva de seguridad de pods

Las directivas de seguridad de pods (Pod Security Policy, PSP) de Kubernetes son recursos del nivel del clúster que controlan la seguridad de los pods. Las PSP le permiten controlar los tipos de pods que se pueden implementar y los tipos de cuentas que pueden implementarlos.

Un recurso PodSecurityPolicy define un conjunto de condiciones que un pod debe cumplir para que pueda implementarse. Si no se cumplen las condiciones, el pod no puede implementarse. Un único recurso PodSecurityPolicy debe validar un pod por completo. Algunas de las reglas de un pod no pueden estar en una directiva y algunas de ellas en otra. Para obtener más información, consulte Directivas de seguridad de pods y RBAC en la documentación de Kubernetes.

Existen varias formas de implementar el uso de directivas de seguridad de pods en Kubernetes. El método habitual consiste en usar objetos de control de acceso basado en funciones (Role-Based Access Control, RBAC). ClusterRole y ClusterRoleBinding se aplican en todo el clúster, mientras que Role y RoleBinding se aplican en un espacio de nombres específico. Si se utiliza RoleBinding, solo permite que los pods se ejecuten en el mismo espacio de nombres que el enlace. Para obtener más información, consulte RBAC en la documentación de Kubernetes.

Los pods de Kubernetes pueden crearse de forma directa o indirecta. Un pod se crea de forma directa mediante la implementación de una especificación de pod con la cuenta de usuario. Para crear un pod de forma indirecta, se define un recurso de nivel superior (como Deployment o DaemonSet). En este caso, una cuenta de servicio crea el pod subyacente. Para obtener más información, consulte Cuentas de servicio en la documentación de Kubernetes.

Para utilizar las PSP de forma efectiva, debe tener en cuenta ambos flujos de trabajo de creación de pods: directo e indirecto. Si un usuario crea un pod de forma directa, la PSP enlazada a la cuenta de usuario controla la operación. Si un usuario crea un pod por medio de una cuenta de servicio, la PSP debe estar enlazada a la cuenta de servicio que se utiliza para crear el pod. Si no se define ninguna cuenta de servicio en la especificación del pod, se utiliza la cuenta de servicio predeterminada del espacio de nombres.

## PodSecurityPolicy predeterminada para clústeres TKG

En la tabla se enumeran y se describen las directivas de seguridad de pods predeterminadas restringidas y con privilegios para los clústeres de TKG, así como el objeto ClusterRole predeterminado que se asocia a cada directiva.

**Tabla 18-4. Instancia de PodSecurityPolicy predeterminada con objeto ClusterRole asociado**

| PSP predeterminada | Permiso | Descripción | Objeto ClusterRole predeterminado asociado |
| --- | --- | --- | --- |
| `vmware-system-privileged` | **Ejecutar como cualquiera** | PSP permisiva. Equivale a la ejecución de un clúster sin la controladora de admisión de PSP habilitada. | `psp:vmware-system-privileged` puede utilizar esta PSP |
| `vmware-system-restricted` | **Debe ejecutarse como no raíz** | PSP restrictiva. No permite el acceso con privilegios a los contenedores de pods, bloquea las posibles escalaciones en la raíz y requiere el uso de varios mecanismos de seguridad. | `psp:vmware-system-restricted` puede utilizar esta PSP |

## Enlaces de función y ClusterRole para clústeres de TKG

TKG en Supervisor no ofrece objetos RoleBinding ni ClusterRoleBinding predeterminados para los clústeres de TKG. En esta documentación se proporcionan ejemplos que puede utilizar. Consulte Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG.

Un usuario de vCenter Single Sign-On al que se otorga el permiso **Editar** en un espacio de nombres de vSphere se asigna a la función **cluster-admin** para cualquier clúster de Tanzu Kubernetes implementado en ese espacio de nombres. Un administrador de clústeres autenticado puede usar implícitamente la PSP de `vmware-system-privileged`. Aunque técnicamente no es un ClusterRoleBinding, tiene el mismo efecto.

El administrador de clústeres debe definir los enlaces para permitir o restringir los tipos de pods que los usuarios pueden implementar en un clúster. Si se utiliza un RoleBinding, el enlace solo permite que los pods se ejecuten en el mismo espacio de nombres que el enlace. Esto puede emparejarse con los grupos del sistema para conceder acceso a todos los pods que se ejecutan en el espacio de nombres. Los usuarios que no son administradores y que se autentican en el clúster se asignan a la función `authenticated` y se pueden enlazar a la PSP predeterminada como tal.

El siguiente comportamiento se aplica a cualquier clúster de TKG que requiera una directiva de seguridad de pods:

- Un administrador de clústeres puede crear pods con privilegios directamente en cualquier espacio de nombres mediante su cuenta de usuario.
- Un administrador de clústeres puede crear implementaciones, StatefulSets y DaemonSet (cada uno de los cuales crea pods con privilegios) en el espacio de nombres kube-system. Si desea utilizar un espacio de nombres de Kubernetes diferente, cree un objeto RoleBinding para él o un objeto ClusterRoleBinding.
- Un administrador de clústeres puede crear sus propias PSP (además de las dos PSP predeterminadas) y enlazar estas PSP a cualquier usuario. Si define su propia PSP, consulte Orden de directivas en la documentación de Kubernetes.
- Ningún usuario autenticado puede crear pods con privilegios o sin privilegios hasta que el administrador del clúster enlaza PSP a los usuarios autenticados.

Para ver ejemplos de enlaces, consulte Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG.

# Aplicar la directiva de seguridad de pods predeterminada a clústeres de servicio TKG

Los clústeres de servicio TKG que utilizan TKR 1.24 y versiones anteriores incluyen la directiva de seguridad de pods predeterminada a la que se puede enlazar para la implementación de cargas de trabajo con privilegios y restringidas.

# Aplicar la directiva de seguridad de pods predeterminada mediante enlaces de función y ClusterRole

vSphere IaaS control plane proporciona una Configurar PSP para TKR 1.24 y versiones anteriores que se puede aplicar a los clústeres de TKG en Supervisor y que funcionan en TKR 1.24 y versiones anteriores. Para ello, cree los objetos RoleBinding y ClusterRoleBinding que hagan referencia a la Configurar PSP para TKR 1.24 y versiones anteriores.

**Nota** Consulte la documentación de Kubernetes para crear su propia directiva de seguridad de pods.

Un objeto RoleBinding concede permisos dentro de un espacio de nombres específico. Un objeto ClusterRoleBinding concede permisos en todo el clúster. La decisión de utilizar RoleBindings o ClusterRoleBinding depende de cada caso práctico. Por ejemplo, si usa ClusterRoleBinding y configura los asuntos para que usen `system:serviceaccounts:<namespace>`, puede enlazar a una PSP antes de que se cree el espacio de nombres. Para obtener más información, consulte RoleBinding y ClusterRoleBinding en la documentación de Kubernetes.

En las siguientes secciones, se brindan comandos de YAML y CLI para crear objetos RoleBinding y ClusterRoleBinding que autorizan el uso de la Configurar PSP para TKR 1.24 y versiones anteriores.

## Ejemplo 1: ClusterRoleBinding para ejecutar un conjunto privilegiado de cargas de trabajo

El siguiente comando kubectl crea un ClusterRoleBinding que otorga acceso a los usuarios autenticados para que ejecuten un conjunto de cargas de trabajo con privilegios mediante la PSP predeterminada `vmware-system-privileged`.

**Advertencia** La aplicación del Ejemplo 1, de forma declarativa o imperativa, permite la implementación de cargas de trabajo con privilegios en todo el clúster. En efecto, el Ejemplo 1 deshabilita los controles de seguridad nativos y debe utilizarse con precaución y con total conocimiento de las implicaciones. Para una seguridad más estricta, considere los Ejemplos 2, 3 y 4.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: psp:privileged
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - vmware-system-privileged
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```

```
  name: all:psp:privileged
roleRef:
  kind: ClusterRole
  name: psp:privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:serviceaccounts
  apiGroup: rbac.authorization.k8s.io
```

Como alternativa a la aplicación de YAML, puede ejecutar el siguiente de comando de `kubectl`:

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

## Ejemplo 2: RoleBinding para ejecutar un conjunto de cargas de trabajo con privilegios

El siguiente comando kubectl crea un RoleBinding que otorga acceso a todas las cuentas de servicio dentro del espacio de nombres predeterminado para ejecutar un conjunto de cargas de trabajo con privilegios mediante la PSP predeterminada `vmware-system-privileged`.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-default-privileged-sa-ns_default
  namespace: default
roleRef:
  kind: ClusterRole
  name: psp:vmware-system-privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

Como alternativa a la aplicación de YAML, puede ejecutar el siguiente de comando de kubectl:

```
kubectl create rolebinding rolebinding-default-privileged-sa-ns_default --namespace=default --
clusterrole=psp:vmware-system-privileged --group=system:serviceaccounts
```

## Ejemplo 3: ClusterRoleBinding para ejecutar un conjunto restringido de cargas de trabajo

El siguiente YAML crea un ClusterRoleBinding que otorga a los usuarios autenticados acceso en todo el clúster para ejecutar un conjunto restringido de cargas de trabajo mediante la PSP predeterminada `vmware-system-restricted`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:authenticated
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

Como alternativa a la aplicación de YAML, puede ejecutar el siguiente de comando de kubectl:

```
kubectl create clusterrolebinding psp:authenticated --clusterrole=psp:vmware-system-
restricted --group=system:authenticated
```

## Ejemplo 4: RoleBinding para ejecutar un conjunto restringido de cargas de trabajo

El siguiente YAML crea un RoleBinding que otorga acceso a todas las cuentas de servicio dentro de un espacio de nombres específico para ejecutar un conjunto restringido de cargas de trabajo mediante la PSP predeterminada `vmware-system-restricted`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:serviceaccounts
  namespace: some-namespace
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

Como alternativa a la aplicación de YAML, puede ejecutar el siguiente de comando de kubectl:

**Nota** Cambie "some-namespace" al espacio de nombres de destino.

```
kubectl create rolebinding psp:serviceaccounts --clusterrole=psp:vmware-system-restricted --
group=system:serviceaccounts -n some-namespace
```

## Función de ejemplo para la directiva de seguridad de pods

Si define su propia directiva de seguridad de pods (Pod Security Policy, PSP) para implementar cargas de trabajo, consulte este ejemplo para crear una función o ClusterRole que haga referencia a la PSP personalizada.

El siguiente ejemplo demuestra una función enlazada a PodSecurityPolicy. En la definición de la función, la función `example-role` otorgada al verbo `use` para un recurso de PSP personalizado que usted defina. De forma alternativa, utilice una de las PSP predeterminadas. A continuación, cree un enlace.

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
  name: example-role
  namespace: tkgs-cluster-ns
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - create
  - get
  - list
  - watch
  - update
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
  - extensions
  resourceNames:
  - CUSTOM-OR-DEFAULT-PSP
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

# Administrar certificados TLS para clústeres del servicio TKG

vSphere IaaS control plane utiliza el cifrado de seguridad de la capa de transporte (Transport Layer Security, TLS) para proteger las comunicaciones entre los componentes. TKG en Supervisor incluye varios certificados TLS que admiten esta infraestructura criptográfica. La rotación de certificados de Supervisor es manual. La rotación de certificados de TKG está automatizada, pero se puede hacer manualmente si es necesario.

## Acerca de los certificados TLS para clústeres del servicio TKG

vSphere IaaS control plane utiliza certificados TLS para proteger la comunicación entre los siguientes componentes:

- vCenter Server
- Nodos del plano de control de Supervisor
- Los hosts ESXi funcionan como nodos de trabajo para los pods de vSphere
- Nodos del clúster de TKG, tanto del plano de control como de trabajo

vSphere IaaS control plane funciona en los siguientes dominios de confianza para autenticar a los usuarios del sistema.

| Dominio de confianza | Descripción |
| --- | --- |
| Dominio de confianza de vCenter | El firmante predeterminado de los certificados TLS en este dominio de confianza es la instancia de VMware Certificate Authority (VMCA) que está integrada en vCenter Server. |
| Dominio de confianza de Kubernetes | El firmante predeterminado de los certificados TLS en este dominio de confianza es la entidad de certificación (Certificate Authority, CA) de Kubernetes. |

Consulte el artículo de la base de conocimientos Guía de certificación de vSphere with Tanzu para conocer más detalles y las listas completas de cada certificado TLS que se utiliza en el entorno de vSphere IaaS control plane.

## Rotación de certificados TLS

El procedimiento para rotar certificados TLS varía en función de si los certificados se utilizan para Supervisor o para clústeres del servicio TKG.

### Rotación de certificados de supervisor

Los certificados TLS para Supervisor se derivan del certificado de VMCA. Consulte el artículo de la base de conocimientos Guía de certificación de vSphere with Tanzu para ver más detalles sobre los certificados de Supervisor.

La rotación de certificados para Supervisor es manual. Consulte el artículo de la base de conocimientos Reemplazar los certificados de supervisor para vSphere with Tanzu si desea obtener instrucciones sobre cómo reemplazar los certificados de Supervisor mediante la herramienta del administrador de certificados de WCP.

### Rotación de certificados del clúster de TKG 2.0

Por lo general, no es necesario rotar manualmente los certificados TLS de un clúster de TKG, ya que cuando se actualiza un clúster de TKG, el proceso de actualización gradual se encarga de rotar automáticamente los certificados TLS.

Si los certificados TLS de un clúster de TKG no han caducado y necesita rotar manualmente estos certificados, puede hacerlo. Para ello, debe completar los pasos que se muestran en la siguiente sección.

## Rotar manualmente los certificados TLS para un clúster del servicio TKG

En estas instrucciones se da por hecho que se tiene experiencia y conocimientos avanzados de la administración de clústeres de TKG. Además, en estas instrucciones se supone que los certificados TLS no caducaron. Si los certificados están caducados, no lleve a cabo estos pasos.

1 Para ejecutar estos pasos, utilice SSH en uno de los nodos del Supervisor. Consulte Conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes.

2 Obtenga el nombre del clúster de TKG.

```
export CLUSTER_NAMESPACE="tkg-cluster-ns"

kubectl get clusters -n $CLUSTER_NAMESPACE
NAME                    PHASE         AGE   VERSION
tkg-cluster             Provisioned   43h
```

3 Obtenga el clúster de TKG kubeconfig.

```
export CLUSTER_NAME="tkg-cluster"

kubectl get secrets -n $CLUSTER_NAMESPACE $CLUSTER_NAME-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > $CLUSTER_NAME-kubeconfig
```

4 Obtenga la clave SSH del clúster de TKG.

```
kubectl get secrets -n $CLUSTER_NAMESPACE $CLUSTER_NAME-ssh -o jsonpath='{.data.ssh-
privatekey}' | base64 -d > $CLUSTER_NAME-ssh-privatekey
chmod 600 $CLUSTER_NAME-ssh-privatekey
```

5 Compruebe el entorno antes de la rotación de certificados.

```
export KUBECONFIG=$CLUSTER_NAME-kubeconfig
```

```
kubectl get nodes -o wide
```

```
kubectl get nodes \
-o jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}' \
-l node-role.kubernetes.io/master= > nodes
```

```
for i in `cat nodes`; do
    printf "\n######\n"
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i hostname
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i sudo kubeadm certs check-expiration
done;
```

Ejemplo de resultado de los comandos anteriores:

```
tkg-cluster-control-plane-k8bqh
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n kube-system get
cm kubeadm-config -o yaml'

CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE
AUTHORITY   EXTERNALLY MANAGED
admin.conf                 Oct 04, 2023 23:00 UTC
363d                                 no
apiserver                  Oct 04, 2023 23:00 UTC   363d
ca                      no
apiserver-etcd-client      Oct 04, 2023 23:00 UTC   363d            etcd-
ca                 no
apiserver-kubelet-client   Oct 04, 2023 23:00 UTC   363d
ca                      no
controller-manager.conf    Oct 04, 2023 23:00 UTC
363d                                 no
etcd-healthcheck-client    Oct 04, 2023 23:00 UTC   363d            etcd-
ca                 no
etcd-peer                  Oct 04, 2023 23:00 UTC   363d            etcd-
ca                 no
etcd-server                Oct 04, 2023 23:00 UTC   363d            etcd-
ca                 no
front-proxy-client         Oct 04, 2023 23:00 UTC   363d            front-proxy-
ca          no
scheduler.conf             Oct 04, 2023 23:00 UTC
363d                                 no

CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED
ca                      Oct 01, 2032 22:56 UTC   9y              no
etcd-ca                 Oct 01, 2032 22:56 UTC   9y              no
front-proxy-ca          Oct 01, 2032 22:56 UTC   9y              no
```

6 Rote los certificados TLS para el clúster de TKG 2.0.

Cambie el contexto al Supervisor antes de continuar con los siguientes pasos.

```
unset KUBECONFIG
kubectl config current-context
kubernetes-admin@kubernetes
```

```
kubectl get kcp  -n $CLUSTER_NAMESPACE $CLUSTER_NAME-control-plane -o
jsonpath='{.apiVersion}{"\n"}'
controlplane.cluster.x-k8s.io/v1beta1
```

```
kubectl get kcp -n $CLUSTER_NAMESPACE $CLUSTER_NAME-control-plane
NAME                        CLUSTER       INITIALIZED   API SERVER AVAILABLE   REPLICAS
READY   UPDATED   UNAVAILABLE   AGE   VERSION
tkg-cluster-control-plane   tkg-cluster   true          true                   3
3       3         0             43h   v1.21.6+vmware.1
```

```
kubectl patch kcp $CLUSTER_NAME-control-plane -n $CLUSTER_NAMESPACE --type merge -p
"{\"spec\":{\"rolloutAfter\":\"`date +'%Y-%m-%dT%TZ'`\"}}"
kubeadmcontrolplane.controlplane.cluster.x-k8s.io/tkg-cluster-control-plane patched
```

Se inició la implementación de la máquina:

```
kubectl get machines -n $CLUSTER_NAMESPACE
NAME                                              CLUSTER
NODENAME
PROVIDERID                                       PHASE          AGE   VERSION
tkg-cluster-control-plane-k8bqh                   tkg-cluster   tkg-cluster-control-plane-
k8bqh                  vsphere://420a2e04-cf75-9b43-f5b6-23ec4df612eb   Running
43h   v1.21.6+vmware.1
tkg-cluster-control-plane-l7hwd                   tkg-cluster   tkg-cluster-control-plane-
l7hwd                  vsphere://420a57cd-a1a0-fec6-a741-19909854feb6   Running
43h   v1.21.6+vmware.1
tkg-cluster-control-plane-mm6xj                   tkg-cluster   tkg-cluster-control-plane-
mm6xj                  vsphere://420a67c2-ce1c-aacc-4f4c-0564daad4efa   Running
43h   v1.21.6+vmware.1
tkg-cluster-control-plane-nqdv6                   tkg-
cluster
                  Provisioning   25s   v1.21.6+vmware.1
tkg-cluster-workers-v8575-59c6645b4-wvnlz         tkg-cluster   tkg-cluster-workers-
v8575-59c6645b4-wvnlz         vsphere://420aa071-9ac2-02ea-6530-eb59ceabf87b
Running        43h   v1.21.6+vmware.1
```

Se completó la implementación de la máquina:

```
kubectl get machines -n $CLUSTER_NAMESPACE
NAME                                              CLUSTER
NODENAME
PROVIDERID                                       PHASE     AGE   VERSION
tkg-cluster-control-plane-m9745                   tkg-cluster   tkg-cluster-control-plane-
m9745                  vsphere://420a5758-50c4-3172-7caf-0bbacaf882d3   Running   17m
v1.21.6+vmware.1
```

```
tkg-cluster-control-plane-nqdv6                   tkg-cluster   tkg-cluster-control-plane-
nqdv6                   vsphere://420ad908-00c2-4b9b-74d8-8d197442e767   Running   22m
v1.21.6+vmware.1
tkg-cluster-control-plane-wdmph                   tkg-cluster   tkg-cluster-control-plane-
wdmph                   vsphere://420af38a-f9f8-cb21-e05d-c1bcb6840a93   Running   10m
v1.21.6+vmware.1
tkg-cluster-workers-v8575-59c6645b4-wvnlz         tkg-cluster   tkg-cluster-workers-
v8575-59c6645b4-wvnlz         vsphere://420aa071-9ac2-02ea-6530-eb59ceabf87b   Running
43h   v1.21.6+vmware.1
```

7 Compruebe la rotación manual de certificados para el clúster de TKG 2.0.

Ejecute los siguientes comandos para comprobar la rotación de certificados:

```
export KUBECONFIG=$CLUSTER_NAME-kubeconfig

kubectl get nodes -o wide
NAME                                        STATUS   ROLES                    AGE
VERSION            INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                 KERNEL-
VERSION       CONTAINER-RUNTIME
tkg-cluster-control-plane-m9745             Ready    control-plane,master   15m
v1.21.6+vmware.1   10.244.0.55   <none>        VMware Photon OS/Linux   4.19.198-1.ph3-
esx   containerd://1.4.11
tkg-cluster-control-plane-nqdv6             Ready    control-plane,master   21m
v1.21.6+vmware.1   10.244.0.54   <none>        VMware Photon OS/Linux   4.19.198-1.ph3-
esx   containerd://1.4.11
tkg-cluster-control-plane-wdmph             Ready    control-plane,master   9m22s
v1.21.6+vmware.1   10.244.0.56   <none>        VMware Photon OS/Linux   4.19.198-1.ph3-
esx   containerd://1.4.11
tkg-cluster-workers-v8575-59c6645b4-wvnlz   Ready    <none>                 43h
v1.21.6+vmware.1   10.244.0.51   <none>        VMware Photon OS/Linux   4.19.198-1.ph3-
esx   containerd://1.4.11

kubectl get nodes \
-o jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}' \
-l node-role.kubernetes.io/master= > nodes

for i in `cat nodes`; do
    printf "\n######\n"
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i hostname
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i sudo kubeadm certs check-expiration
done;
```

Ejemplo de resultado donde se muestran las fechas de caducidad actualizadas.

```
######
tkg-cluster-control-plane-m9745
[check-expiration] Reading configuration from the cluster...
[check-expiration] FYI: You can look at this config file with 'kubectl -n kube-system get
cm kubeadm-config -o yaml'

CERTIFICATE                EXPIRES                  RESIDUAL TIME   CERTIFICATE
AUTHORITY   EXTERNALLY MANAGED
```

```
admin.conf                 Oct 06, 2023 18:18 UTC
364d                                    no
apiserver                  Oct 06, 2023 18:18 UTC   364d
ca                      no
apiserver-etcd-client      Oct 06, 2023 18:18 UTC   364d            etcd-
ca                 no
apiserver-kubelet-client   Oct 06, 2023 18:18 UTC   364d
ca                      no
controller-manager.conf    Oct 06, 2023 18:18 UTC
364d                                    no
etcd-healthcheck-client    Oct 06, 2023 18:18 UTC   364d            etcd-
ca                 no
etcd-peer                  Oct 06, 2023 18:18 UTC   364d            etcd-
ca                 no
etcd-server                Oct 06, 2023 18:18 UTC   364d            etcd-
ca                 no
front-proxy-client         Oct 06, 2023 18:18 UTC   364d            front-proxy-
ca          no
scheduler.conf             Oct 06, 2023 18:18 UTC
364d                                    no

CERTIFICATE AUTHORITY   EXPIRES                  RESIDUAL TIME   EXTERNALLY MANAGED
ca                      Oct 01, 2032 22:56 UTC   9y              no
etcd-ca                 Oct 01, 2032 22:56 UTC   9y              no
front-proxy-ca          Oct 01, 2032 22:56 UTC   9y              no
```

8 Compruebe el certificado de Kubelet.

No es necesario rotar el certificado de Kubelet si se da por hecho que el parámetro `rotateCertificates` en la configuración de kubelet está establecido en `true`, que es la configuración predeterminada.

Esta configuración se puede verificar mediante los siguientes comandos:

```
kubectl get nodes \
-o jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}' \
-l node-role.kubernetes.io/master!= > workernodes

for i in `cat workernodes`; do
    printf "\n######\n"
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i hostname
    ssh -o "StrictHostKeyChecking=no" -i $CLUSTER_NAME-ssh-privatekey -q vmware-system-
user@$i sudo grep rotate /var/lib/kubelet/config.yaml
done;
```

Resultado de ejemplo:

```
######
tkg-cluster-workers-v8575-59c6645b4-wvnlz
rotateCertificates: true
```

# Rotar certificados de NSX

Supervisor utiliza TLS para la comunicación entre el Supervisor y NSX. Es posible que tenga que rotar varios certificados de NSX si implementó el Supervisor con la pila de redes NSX.

## Certificados de NSX utilizados por Supervisor

WCP con NSX utiliza dos certificados para la integración con NSX:

- Certificado y clave del equilibrador de carga de NSX
- Certificado y clave de NSX Manager

Para obtener más información sobre estos certificados, consulte Certificados de la federación de NSX en la *Guía de administración de NSX*.

**Nota** La información de este tema se basa en NSX v3.2.

## Rotar la clave y el certificado del equilibrador de carga de NSX

Es posible rotar la clave y el certificado TLS del equilibrador de carga de NSX en la pantalla **Supervisor** > **Certificados** > **Equilibrador de carga de NSX**.

- Seleccione **Acciones** > **Generar CSR** para generar el certificado.
- Seleccione **Acciones** > **Reemplazar certificado** para actualizar el certificado y la clave.

## Generar una clave y un certificado autofirmado para cada nodo de NSX Manager

Supervisor utiliza la cuenta de administrador de empresa para acceder a la API de NSX Manager. Si el certificado de NSX Manager caduca, el Supervisor no podrá acceder a NSX.

Si caducan los certificados de NSX Manager, compruebe el registro de Supervisor:

```
tail -f /var/log/vmware/wcp/wcpsvc.log
```

Es posible que aparezcan errores similares al siguiente:

```
error wcp [kubelifecycle/nsx_pi.go:47] ... Error creating WCP service principal identity.
Err: NSX service-wide principal identity creation failed: ... x509: certificate has expired
```

```
error wcp [kubelifecycle/controller.go:554] ... Failed to create WCP service PI in NSX.
Err: WCP service principal idenitity creation failed: NSX service-wide principal identity
creation failed:
... x509: certificate has expired
```

Para solucionar el problema, actualice el certificado y la clave de cada nodo de NSX Manager. Si utiliza un clúster de administración de NSX de 3 nodos con una dirección VIP, tenga en cuenta que Supervisor no utiliza la dirección VIP. Esto significa que tendrá que rotar cada certificado en cada uno de los nodos de NSX Manager. Para rotar los certificados no basta únicamente con reemplazar el certificado VIP.

1 Para rotar el certificado de un nodo de NSX Manager, cree una solicitud de firma de certificado denominada y rellénela con el contenido que aparece a continuación.

   Donde:

   - `NSX-MGR-IP-ADDRESS` es la dirección IP de NSX Manager
   - `NSX-MGR-FQDN` es el FQDN o la dirección IP de NSX Manager

   `nsx-mgr-01-cert.cnf`

```
[ req ]
default_bits = 2048
default_md = sha256
prompt = no
distinguished_name = req_distinguished_name
x509_extensions = SAN
req_extensions = v3_ca

[ req_distinguished_name ]
countryName = US
stateOrProvinceName = California
localityName = CA
organizationName = NSX
commonName = NSX-MGR-IP-ADDRESS  #CAN ONLY USE IF SAN IS ALSO USED

[ SAN ]
basicConstraints = CA:false
subjectKeyIdentifier = hash
authorityKeyIdentifier=keyid:always,issuer:always

[ v3_ca ]
subjectAltName = DNS:NSX-MGR-FQDN,IP:NSX-MGR-IP-ADDRESS  #MUST USE
```

   Por ejemplo:

```
[ req ]
default_bits = 2048
default_md = sha256
prompt = no
distinguished_name = req_distinguished_name
x509_extensions = SAN
req_extensions = v3_ca

[ req_distinguished_name ]
countryName = US
stateOrProvinceName = California
localityName = CA
```

```
organizationName = NSX
commonName = 10.197.79.122

[ SAN ]
basicConstraints = CA:false
subjectKeyIdentifier = hash
authorityKeyIdentifier=keyid:always,issuer:always

[ v3_ca ]
subjectAltName = DNS:10.197.79.122,IP:10.197.79.122
```

2 Utilice OpenSSL para generar el certificado SSL y la clave privada.

```
openssl req -newkey rsa -nodes -days 1100 -x509 -config nsx-mgr-01-cert.cnf -keyout nsx-
mgr-01.key -out nsx-mgr-01.crt
```

3 Compruebe que aparece el siguiente resultado después de ejecutar el comando.

```
Generating a 2048 bit RSA private key
...............+++
................+++
writing new private key to 'nsx-mgr-01.key'
```

4 Debería ver 3 archivos: la solicitud de firma inicial y el certificado y la clave privada generados al ejecutar la solicitud de firma.

5 Ejecute el siguiente comando para comprobar el certificado y la clave privada.

```
openssl x509 -in nsx-mgr-01.crt -text -noout
```

6 Si utiliza un clúster de administración de NSX de varios nodos, repita el proceso para cada nodo de NSX Manager. Cambie la dirección IP y el FQDN en la solicitud de firma del certificado, y los nombres del archivo de salida según corresponda.

## Importar el certificado SSL y la clave privada a la consola de administración de NSX-T

Importe cada certificado de nodo de NSX Manager y cada clave privada en NSX con los pasos siguientes. Si guarda los archivos `nsx.crt` y `nsx.key` de forma local, puede cargarlos en NSX o puede copiar y pegar el contenido.

1 Inicie sesión en la consola de administración de NSX y desplácese hasta la página **Sistema > Certificados**.

2 Haga clic en **Importar > Importar certificado**.

**Nota** Dado que generó un certificado autofirmado, asegúrese de seleccionar **Importar certificado** y no **Importar certificado de CA**.

3 Introduzca un **nombre** descriptivo para el certificado y el par de claves, como `nsx-mgr-01-cert-and-key`.

4 Busque y seleccione el archivo de certificado, o copie y pegue su contenido, incluidos el encabezado y el pie de página.

Por ejemplo:

```
-----BEGIN CERTIFICATE-----
MIID+zCCAuOgAwIBAgIUCfXaWxNwXvrEFQbt+Dvvp9C/UkIwDQYJKoZIhvcNAQEL
BQAwVTELMAkGA1UEBhMCVVMxEzARBgNVBAgMCkNhbGlmb3JuaWExCzAJBgNVBAcM
...
FGlnyT4vxpa2TxvXNTCuXPV9z0VtVBF2QpUJluGH7W1i2wUnApCCXhItcBkfve0f
pCi9YoRoUT8fuMBYo7sL
-----END CERTIFICATE-----
```

5 Busque y seleccione la clave, o copie y pegue su contenido, incluidos el encabezado y el pie de página.

Por ejemplo:

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQC5GNlUSYHa1p+E
XuOGAsIgiFxUnerRYNm2ARMqRb/xKK6R4xgZhBmpmikpE90vQibvouHqnL13owq7
...
OzbnwMCUI2TeY1iJNx3HNKUrdLvrr8CMh7Looe0L4/2j7ygew2x2C5m272SCJYs/
ly+bOXEYaH4/ORHbvvr0jQ==
-----END PRIVATE KEY-----
```

6 Seleccione **No** para la opción **Certificado de servicio**.

7 Introduzca una descripción para el certificado y el par de claves, como `Cert and Private Key for NSX Manager Node 1`.

8 Haga clic en **Importar**.

9 Repita el proceso para cada uno de los certificados de NSX Manager y el par de claves.

## Registrar el certificado de NSX Manager mediante la NSX API

Una vez que haya cargado los certificados y las claves en NSX Manager, regístrelos mediante la NSX API. Consulte también Importar y reemplazar certificados en la *Guía de administración de NSX*.

1 En NSX Manager, seleccione **Sistema > Certificados**.

2 En la columna de identificadores, seleccione el identificador del certificado que desee registrar y cópielo de la ventana emergente.

3 Ejecute la siguiente llamada API para enumerar los certificados. Obtenga los identificadores de nodo de certificado de cada certificado que desee actualizar.

```
GET https://NSX-MGR-IP-or-FQDN/api/v1/trust-management/certificates
```

4 Ejecute la siguiente llamada API para validar el certificado.

```
GET https://NSX-MGR-IP-or-FQDN/api/v1/trust-management/certificates/<cert-id>?
action=validate
```

Por ejemplo:

```
https://10.19.92.133/api/v1/trust-management/certificates/070bae44-7548-45ff-
a884-578f079eb6d4?action=validate
```

5 Ejecute la siguiente llamada API para reemplazar el certificado de un nodo de NSX Manager:

```
POST https://NSX-MGR-IP-or-FQDN/api/v1/trust-management/certificates/<cert-id>?
action=apply_certificate&service_type=API&node_id=<node-id>
```

Por ejemplo:

```
POST https://10.19.92.133/api/v1/trust-management/certificates/070bae44-7548-45ff-
a884-578f079eb6d4?
action=apply_certificate&service_type=API&node_id=e61c7537-3090-4149-b2b6-19915c20504f
```

6 Si utiliza un clúster de administración de NSX de varios nodos, repita el proceso de reemplazo de certificados para cada nodo de NSX Manager.

7 Cuando haya terminado, elimine cada certificado caducado que reemplazó. Puede hacerlo mediante la interfaz de NSX Manager o con la NSX API.

Por ejemplo:

```
https://NSX-MGR-IP-or-FQDN/api/v1/trust-management/certificates/<cert-id>
```

# Integrar TMC con clústeres de Servicio TKG

19

En esta sección, se tratan temas para la integración de Tanzu Mission Control con clústeres de Servicio TKG.

Lea los siguientes temas a continuación:



## Registrar Tanzu Mission Control alojado con Supervisor

Es posible integrar Tanzu Kubernetes Grid en el Supervisor con Tanzu Mission Control. Si lo hace, podrá administrar clústeres de Servicio TKG mediante la interfaz web de TMC.

### Uso de clústeres de Servicio TKG con Tanzu Mission Control

El flujo de trabajo general para el uso de clústeres de Servicio TKG con Tanzu Mission Control y Tanzu Application Platform es el siguiente:

| Paso | Acción |
|---|---|
| 1 | Aprovisionar un clúster de carga de trabajo de TKG<br>Consulte Capítulo 7 Aprovisionar clústeres del servicio de TKG. |
| 2 | Registrar Supervisor con TMC<br>Consulte Registrar Supervisor con Tanzu Mission Control. |
| 3 | Asociar el clúster de carga de trabajo de TKG a TMC<br>Consulte Administrar el ciclo de vida de los clústeres de Tanzu Kubernetes |
| 4 | Instalar Tanzu Application Platform<br>Consulte la documentación de Tanzu Application Platform. |

### Registrar Supervisor con Tanzu Mission Control

El Supervisor se envía con un espacio de nombres de vSphere dedicado para Tanzu Mission Control. Para registrar Supervisor con Tanzu Mission Control, instale el agente de TMC en el espacio de nombres de vSphere y complete el proceso de registro mediante TMC.

Complete este procedimiento para instalar el agente de TMC en Supervisor. Tenga en cuenta que este procedimiento requiere que se utilice vSphere Client y la interfaz web de TMC de forma simultánea.

1 Si utiliza el complemento de vSphere para kubectl, autentíquese en el Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 Cambie el contexto al Supervisor, por ejemplo:

```
kubectl config use-context 10.199.95.59
```

3 Ejecute el siguiente comando para enumerar los espacios de nombres.

```
kubectl get ns
```

4 El espacio de nombres de vSphere para TMC se denomina `svc-tmc-cXX` (donde XX es un número). Compruebe que existe y está activo.

5 Inicie sesión en la interfaz web de Tanzu Mission Control.

6 Registre el Supervisor en Tanzu Mission Control. Consulte Registrar un clúster de administración en Tanzu Mission Control.

7 Mediante la interfaz web de Tanzu Mission Control, copie la URL de registro para el Supervisor que se encuentra disponible en la página **Administración > Clústeres de administración**.

8 Si es necesario, abra un puerto de firewall en el Supervisor para el puerto que requiere Tanzu Mission Control (generalmente `443`). Consulte Conexiones salientes realizadas por las extensiones del agente de clúster.

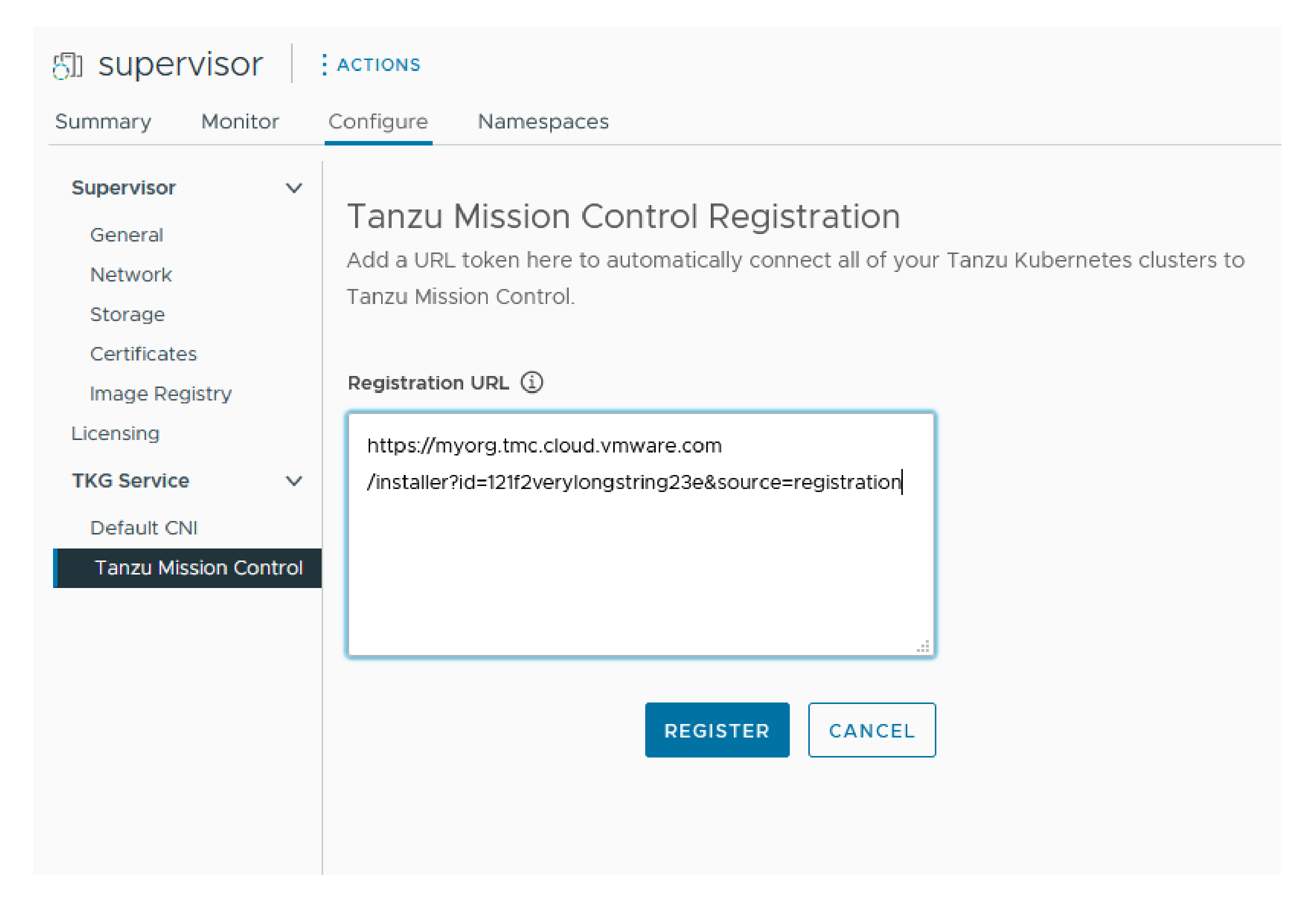
9 Inicie sesión en el entorno de vSphere IaaS control plane mediante vSphere Client.

10 Seleccione **Administración de cargas de trabajo** y el Supervisor de destino.

11 Seleccione **Configurar** y seleccione **Servicio TKG > Tanzu Mission Control**.

12 Pegue la URL de registro en el campo **URL de registro**.

13 Haga clic en **Registrar**.




Después de registrar Supervisor en TMC, utilice la interfaz web de TMC para aprovisionar y administrar los clústeres de TKG. Consulte la documentación de Tanzu Mission Control para obtener instrucciones.

## Desinstale el agente de Tanzu Mission Control de Supervisor

Si quiere desinstalar el agente de Tanzu Mission Control del Supervisor, consulte Eliminar manualmente el agente de clúster de un clúster supervisor en vSphere with Tanzu.

# Registrar Tanzu Mission Control autoadministrado con Supervisor

Para registrar Tanzu Mission Control autoadministrado con Supervisor, cree y aplique una definición de recursos personalizados para el agente de TMC.

## Acerca de Tanzu Mission Control autoadministrado

Para obtener más información sobre Tanzu Mission Control autoadministrado, incluido cómo instalarlo y configurarlo, consulte la documentación Instalar y ejecutar VMware Tanzu Mission Control autoadministrado.

## Registrar Tanzu Mission Control autoadministrado con Supervisor

Para integrar Tanzu Mission Control autoadministrado con Supervisor, cree una definición de recursos personalizados con referencias al agente de TMC. Supervisor incluye un espacio de nombres de Kubernetes para TMC en el que está instalado el agente.

Complete el siguiente procedimiento.

1 Instale Tanzu Mission Control autoadministrado como se describe en la documentación. Consulte Instalar y ejecutar VMware Tanzu Mission Control autoadministrado.

2 Con un navegador web, acceda a la implementación local de Tanzu Mission Control autoadministrado.

3 Exporte el certificado de CA raíz para la instalación de Tanzu Mission Control autoadministrado.

   - Si utiliza una entidad de certificación conocida, haga clic en el icono de candado que está situado a la izquierda de la barra de direcciones en el navegador y vea el certificado. Si utiliza una entidad de certificación privada, haga clic en el botón No seguro y vea el certificado.
   - En la ventana emergente del certificado, seleccione la pestaña `Details` y, a continuación, el botón `Export` para descargar una copia del certificado de la entidad de certificación.
   - Abra el archivo del certificado de la entidad de certificación con el editor de texto que elija para acceder al contenido del certificado de la CA.

4 Si utiliza el complemento de vSphere para kubectl, autentíquese en el Supervisor.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

5 Ejecute el siguiente comando para ver los contextos de Kubectl disponibles.

```
kubectl config get-contexts
```

6 Cambie el contexto al espacio de nombres de vSphere de destino en el que se aprovisiona el clúster de TKG que ejecuta Tanzu Mission Control autoadministrado.

```
kubectl config use-context VSPHERE-NAMESPACE
```

7 Ejecute el siguiente comando para enumerar los espacios de nombres de Kubernetes.

```
kubectl get ns
```

8 El espacio de nombres de Kubernetes en Supervisor para TMC se denomina `svc-tmc-cXXXX` (donde XXXX es un número). Por ejemplo, `svc-tmc-c1208`. Compruebe que este espacio de nombres de Kubernetes existe y está activo.

9 Utilice un editor de texto para crear la definición de recursos personalizados denominada `agentconfig.yaml`. Este archivo incluye el espacio de nombres de TMC, el nombre de host de la implementación de TMC autoadministrado y el contenido del certificado de la entidad de certificación.

- Introduzca el nombre del espacio de nombres de Kubernetes para TMC en el campo `namespace`.
- Introduzca los certificados de CA en los campos `caCerts`.
- Introduzca el nombre de host de TMC en el campo `allowedHostNames`.

```
apiVersion: "installers.tmc.cloud.vmware.com/v1alpha1"
kind: "AgentConfig"
metadata:
  name: "tmc-agent-config"
  namespace: "<namespace>"
spec:
  caCerts: |-
    -----BEGIN CERTIFICATE-----
    Certificate1
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    Certificate2
    -----END CERTIFICATE-----
  allowedHostNames:
    - "google.com"
```

10 Aplique el archivo YAML `AgentConfig` que creó.

```
kubectl apply -f agentconfig.yaml
```

11 Complete el proceso de registro mediante la interfaz web de su instalación de Tanzu Mission Control autoadministrado. Consulte la documentación de Tanzu Mission Control para obtener instrucciones.

# Copia de seguridad y restauración de cargas de trabajo y clústeres de servicio TKG

20

Consulte esta sección para realizar copias de seguridad y restaurar las cargas de trabajo y los clústeres de servicio TKG.

Lea los siguientes temas a continuación:



## Consideraciones para realizar copias de seguridad y restaurar cargas de trabajo y clústeres de Servicio TKG

En este tema se ofrecen las consideraciones que hay que tener en cuenta para realizar copias de seguridad y restaurar las cargas de trabajo que se ejecutan en clústeres de Servicio TKG.

### Hacer una copia de seguridad y restaurar clústeres de Servicio TKG

Para realizar una copia de seguridad y restaurar clústeres de TKG, realice una copia de seguridad de la base de datos de Supervisor. Si lo hace, podrá restaurar objetos de espacios de nombres de vSphere y máquinas virtuales de nodo de clústeres de TKG.

Habilite la copia de seguridad y restauración de Supervisor mediante la función de copia de seguridad de vCenter Server disponible a través de la interfaz de administración de vCenter Server. Para obtener más información, consulte la publicación de restauración de copia de seguridad de vSphere IaaS control plane.

**Nota** Puede utilizar la copia de seguridad de Supervisor solo para restaurar las máquinas virtuales del nodo de clúster de TKG. No se puede utilizar la copia de seguridad de Supervisor para restaurar las cargas de trabajo implementadas en clústeres de TKG. Debe realizar una copia de seguridad de las cargas de trabajo por separado y, a continuación, restaurarlas después de restaurar el clúster.

## Copia de seguridad y restauración de cargas de trabajo que se ejecutan en clústeres de Servicio TKG

En esta tabla se resumen las opciones para hacer copias de seguridad y restaurar cargas de trabajo sin estado y con estado que se ejecutan en clústeres de TKG.

**Nota** La copia de seguridad y la restauración de un clúster de Kubernetes mediante Velero independiente le ofrece portabilidad. Esto significa que si desea poder restaurar las cargas de trabajo del clúster en un clúster de Kubernetes no aprovisionado por Servicio TKG, debe utilizar Velero independiente.

| Situación | Herramienta | Comentarios |
| --- | --- | --- |
| Haga una copia de seguridad de las cargas de trabajo sin estado y con estado en un clúster de TKG en Supervisor y restaure en un clúster de TKG en Supervisor. | Complemento de Velero para vSphere<br>Consulte Copia de seguridad y restauración de cargas de trabajo mediante complemento de Velero para vSphere. | Se puede realizar una copia de seguridad y restaurar tanto los metadatos de Kubernetes como los volúmenes persistentes.<br>La creación de instantáneas de Velero se utiliza para volúmenes persistentes con aplicaciones con estado.<br>Requiere que el complemento de Velero para vSphere también esté instalado y configurado en supervisor. |
| Haga una copia de seguridad de las cargas de trabajo sin estado y con estado en un clúster de TKG en Supervisor y restaure en un clúster de Kubernetes compatible. | Restic y Velero independientes<br>Consulte Realizar una copia de seguridad y restaurar cargas de trabajo de clústeres de TKG en Supervisor mediante Restic y Velero independientes.<br>. | Se puede realizar una copia de seguridad y restaurar tanto los metadatos de Kubernetes como los volúmenes persistentes.<br>Restic se utiliza para volúmenes persistentes con aplicaciones con estado.<br>Utilice este enfoque si necesita portabilidad. |
| Haga una copia de seguridad de las cargas de trabajo sin estado y con estado en un clúster de TKG en Supervisor y restaure en un clúster de Kubernetes compatible. | Instancia de Velero independiente con instantáneas de CSI<br>Consulte Copia de seguridad y restauración mediante Velero con instantánea de CSI. | Requiere vSphere 8.0 U2+ y TKr v1.26+ para vSphere 8.0. |

# Copia de seguridad y restauración de cargas de trabajo mediante complemento de Velero para vSphere

En esta sección se proporcionan temas para hacer copias de seguridad y restaurar cargas de trabajo de clústeres mediante complemento de Velero para vSphere.

# Instalar y configurar el complemento de Velero para vSphere en un clúster de TKG

Es posible utilizar el complemento de Velero para vSphere para realizar una copia de seguridad y una restauración de las cargas de trabajo que se ejecutan en un clúster de TKGS mediante la instalación del complemento de Velero para vSphere en ese clúster.

## Descripción general

El complemento de Velero para vSphere proporciona una solución para realizar una copia de seguridad y una restauración de las cargas de trabajo de clúster de TKGS. Para cargas de trabajo persistentes, el complemento de Velero para vSphere le permite tomar instantáneas de los volúmenes persistentes.

---

**Nota** Si necesita portabilidad para las cargas de trabajo del clúster de TKGS que desea restaurar y para las que quiere hacer una copia de seguridad, no utilice el complemento de Velero para vSphere. Para la portabilidad entre clústeres de Kubernetes, utilice Velero independiente con Restic.

---

## Requisito previo: instale el complemento de Velero para vSphere en el Supervisor

Para la instalación del complemento de Velero para vSphere en un clúster de TKGS, se requiere que el Supervisor tenga instalado el complemento de Velero para vSphere. Además, Supervisor debe estar configurado con redes NSX. Consulte #unique_22.

## Paso 1: Instalar la CLI de Velero en una Workstation de Linux

La CLI de Velero es la herramienta estándar para interactuar con Velero. La CLI de Velero proporciona más funcionalidad que la CLI del complemento de Velero para vSphere (`velero-vsphere`) y es necesaria para realizar copias de seguridad y restauración de las cargas de trabajo del clúster de Tanzu Kubernetes.

Instale la CLI de Velero en una estación de trabajo Linux. Idealmente, este es el mismo host de salto en el que se ejecutan las CLI asociadas para el entorno de vSphere IaaS control plane, incluidos `kubectl`, `kubectl-vsphere` y `velero-vsphere`.

Los números de versión de Velero se presentan como `X.Y.Z`. Consulte la Matriz de compatibilidad de Velero para ver las versiones específicas que deben utilizarse y sustitúyalas según corresponda al ejecutar los comandos.

Complete los siguientes pasos para instalar la CLI de Velero.

1 Ejecute los siguientes comandos:

```
$ wget https://github.com/vmware-tanzu/velero/releases/download/vX.Y.Z/velero-vX.Y.Z-linux-amd64.tar.gz
$ gzip -d velero-vX.Y.Z-linux-amd64.tar.gz && tar -xvf velero-vX.Y.Z-linux-amd64.tar
$ export PATH="$(pwd)/velero-vX.Y.Z-linux-amd64:$PATH"

$ which velero
/root/velero-vX.Y.Z-linux-amd64/velero
```

2 Compruebe la instalación de la CLI de Velero.

```
velero version

Client:
    Version: vX.Y.Z
```

## Paso 2: Obtener los detalles del depósito compatible con S3

Para mayor comodidad, los pasos asumen que está utilizando el mismo almacén de objetos compatible con S3 que configuró cuando instaló el complemento de Velero para vSphere en el Supervisor. En producción, es posible que desee crear un almacén de objetos independiente.

Para instalar el complemento de Velero para vSphere, deberá proporcionar la siguiente información sobre el almacén de objetos compatible con S3.

| Elemento de datos | Valor de ejemplo |
|---|---|
| s3Url | `http://my-s3-store.example.com` |
| aws_access_key_id | ACCESS-KEY-ID-STRING |
| aws_secret_access_key | SECRET-ACCESS-KEY-STRING |

Cree un nombre de archivo de secretos `s3-credentials` con la siguiente información. Debe hacer referencia a este archivo cuando instale el complemento de Velero para vSphere.

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

## Paso 3 Opción A: Instalar complemento de Velero para vSphere en el clúster de TKG con una etiqueta (nuevo método)

Si utiliza vSphere 8 Update 3 o una versión posterior, puede instalar complemento de Velero para vSphere automáticamente en un clúster de TKG agregando una etiqueta.

1 Compruebe que se pueda acceder a la ubicación de almacenamiento de la copia de seguridad.

2 Compruebe que esté activada la función servicio de supervisor de Velero vSphere Operator Core.

```
kubectl get ns | grep velero
svc-velero-domain-c9                              Active   18d
```

3 Compruebe que se haya creado un espacio de nombres de Kubernetes con el nombre `velero` en Supervisor.

```
kubectl get ns | grep velero
svc-velero-domain-c9                              Active   18d
velero                                            Active   1s
```

4 Compruebe que servicio de supervisor de complemento de Velero para vSphere esté habilitado en Supervisor.

```
velero version
Client:
        Version: v1.11.1
        Git commit: bdbe7eb242b0f64d5b04a7fea86d1edbb3a3587c
Server:
        Version: v1.11.1
```

```
kubectl get veleroservice -A
NAMESPACE   NAME      AGE
velero      default   53m
```

```
velero backup-location get
NAME      PROVIDER   BUCKET/PREFIX   PHASE       LAST VALIDATED                  ACCESS
MODE   DEFAULT
default   aws        velero          Available   2023-11-20 14:10:57 -0800 PST
ReadWrite     true
```

5 Agregue la etiqueta velero al clúster a fin de habilitar Velero para el clúster de TKG de destino.

```
kubectl label cluster CLUSTER-NAME --namespace CLUSTER-NS velero.vsphere.vmware.com/
enabled=true
```

**Nota** Esto se realiza desde espacio de nombres de vSphere cuando se aprovisiona el clúster.

6 Compruebe que Velero esté instalado y listo para el clúster.

```
kubectl get ns
NAME                                 STATUS   AGE
...
velero                               Active   2m    <--
velero-vsphere-plugin-backupdriver   Active   2d23h
```

```
kubectl get all -n velero
NAME                                 READY   STATUS    RESTARTS   AGE
pod/backup-driver-5945d6bcd4-gtw9d   1/1     Running   0          17h
pod/velero-6b9b49449-pq6b4           1/1     Running   0          18h
NAME                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/backup-driver   1/1     1            1           17h
deployment.apps/velero          1/1     1            1           18h
NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/backup-driver-5945d6bcd4   1         1         1       17h
replicaset.apps/velero-6b9b49449           1         1         1       18h
```

```
velero version
Client:
        Version: v1.11.1
        Git commit: bdbe7eb242b0f64d5b04a7fea86d1edbb3a3587c
Server:
        Version: v1.11.1
```

## Paso 3 Opción B: Instalar complemento de Velero para vSphere en el clúster de TKG manualmente (método heredado)

Deberá utilizar la CLI de Velero para instalar el complemento de Velero para vSphere en el clúster de TKG de destino del que desea realizar una copia de seguridad y restauración.

El contexto de la CLI de Velero seguirá automáticamente el contexto de `kubectl`. Antes de ejecutar los comandos de la CLI de Velero para instalar Velero y el complemento de Velero para vSphere en el clúster de destino, asegúrese de establecer el contexto de `kubectl` en el clúster de destino.

1 Utilice el complemento de vSphere para kubectl para autenticarse en el Supervisor.

2 Establezca el contexto de `kubectl` en el clúster de TKG de destino.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

3 En el clúster de TKG, cree un mapa de configuración para el complemento de Velero denominado `velero-vsphere-plugin-config.yaml`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-vsphere-plugin-config
data:
  cluster_flavor: GUEST
```

Aplique el mapa de configuración en el clúster de TKG.

```
kubectl apply -n <velero-namespace> -f velero-vsphere-plugin-config.yaml
```

Si no instala el mapa de configuración, recibirá el siguiente error cuando intente instalar el complemento de Velero para vSphere.

```
Error received while retrieving cluster flavor from config, err: configmaps "velero-
vsphere-plugin-config" not found
Falling back to retrieving cluster flavor from vSphere CSI Driver Deployment
```

4 Ejecute el siguiente comando de la CLI de Velero para instalar Velero en el clúster de destino.

Reemplace los valores de marcador de posición de los campos **`BUCKET-NAME`**, **`REGION`** (dos instancias) y **`s3Url`** con los valores adecuados. Si se desvió de cualquiera de las instrucciones anteriores, ajuste también esos valores, como el nombre o la ubicación del archivo de secretos, el nombre del espacio de nombres `velero` creado manualmente, etc.

```
./velero install --provider aws \
--bucket BUCKET-NAME \
--secret-file ./s3-credentials \
--features=EnableVSphereItemActionPlugin \
--plugins velero/velero-plugin-for-aws:vX.Y.Z \
--snapshot-location-config region=REGION \
--backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

5 Instale el complemento de Velero para vSphere en el clúster de destino. El Velero instalado se comunicará con el servidor de API de Kubernetes para instalar el complemento.

```
velero plugin add vsphereveleroplugin/velero-plugin-for-vsphere:vX.Y.Z
```

## Anexo: Desinstalar el complemento de Velero para vSphere del clúster de TKG

Siga estos pasos para desinstalar el complemento de Velero para vSphere.

1 Establezca el contexto de `kubectl` en el clúster de destino de Tanzu Kubernetes.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

2 Para desinstalar el complemento, ejecute el siguiente comando para eliminar el InitContainer de velero-plugin-for-vsphere de la implementación de Velero.

```
velero plugin remove vsphereveleroplugin/velero-plugin-for-vsphere:vX.Y.Z
```

3 Para completar el proceso, elimine la implementación del controlador de copia de seguridad y los CRD relacionados.

```
kubectl -n velero delete deployment.apps/backup-driver
```

```
kubectl delete crds \
backuprepositories.backupdriver.cnsdp.vmware.com \
backuprepositoryclaims.backupdriver.cnsdp.vmware.com \
clonefromsnapshots.backupdriver.cnsdp.vmware.com \
deletesnapshots.backupdriver.cnsdp.vmware.com \
snapshots.backupdriver.cnsdp.vmware.com
```

```
kubectl delete crds uploads.datamover.cnsdp.vmware.com downloads.datamover.cnsdp.vmware.com
```

# Realizar una copia de seguridad y restaurar cargas de trabajo de clúster de TKG mediante complemento de Velero para vSphere

Puede realizar copias de seguridad y restaurar cargas de trabajo que se ejecutan en clústeres de TKG en Supervisor mediante el complemento de Velero para vSphere.

## Requisitos previos

Para realizar una copia de seguridad y restaurar cargas de trabajo de los clústeres de TKG mediante el complemento de Velero para vSphere, primero debe instalar el complemento de Velero para vSphere en el clúster de destino. Consulte Instalar y configurar el complemento de Velero para vSphere en un clúster de TKG.

## Copia de seguridad de una carga de trabajo

A continuación se muestra un comando de ejemplo para crear una copia de seguridad de Velero.

```
velero backup create <backup name> --include-namespaces=my-namespace
```

La copia de seguridad de Velero se marcará como `Completed` después de que se hayan tomado todas las instantáneas locales y se hayan cargado los metadatos de Kubernetes, excepto las instantáneas de volumen, en el almacén de objetos. En este punto, las tareas de movimiento de datos asincrónicas, es decir, la carga de instantáneas de volumen, aún se están realizando en segundo plano y pueden tardar algún tiempo en completarse. Podemos comprobar el estado de las instantáneas de volumen mediante la supervisión de los recursos personalizados (CR) de instantánea.

## Snapshots

Las instantáneas se utilizan para realizar copias de seguridad de volúmenes persistentes. Para cada instantánea de volumen, se crea un CR de instantánea en el mismo espacio de nombres que la notificación de volumen persistente (PVC) de que se crea una instantánea.

Puede obtener todas las instantáneas en el espacio de nombres de PVC ejecutando el siguiente comando.

```
kubectl get -n <pvc namespace> snapshot
```

La definición de recursos personalizados (CRD) de instantánea tiene varias fases para el campo `.status.phase`, que incluyen lo siguiente:

| Fase de instantánea | Descripción |
| --- | --- |
| Novedad | Aún no procesada |
| Snapshotted | Se tomó una instantánea local |
| SnapshotFailed | Se produjo un error en la instantánea local |
| Uploading | Se está cargando la instantánea |
| Uploaded | Se cargó la instantánea |
| UploadFailed | No se pudo cargar la instantánea |
| Canceling | Se está cancelando la carga de la instantánea |
| Canceled | Se canceló la carga de la instantánea |
| CleanupAfterUploadFailed | Se produjo un error en la limpieza de la instantánea local después de la carga de la instantánea |

## Restaurar una carga de trabajo

A continuación se muestra un comando de ejemplo de restauración de Velero.

```
velero restore create --from-backup <velero-backup-name>
```

La restauración de Velero se marcará como `Completed` cuando las instantáneas de volumen y otros metadatos de Kubernetes se hayan restaurado correctamente en el clúster actual. En este punto, también se completan todas las tareas del complemento de vSphere relacionadas con esta restauración. No hay tareas de movimiento de datos asincrónicas en segundo plano como en el caso de la copia de seguridad de Velero.

## CloneFromSnapshots

Para restaurar desde cada instantánea de volumen, se creará un recurso personalizado (CR) `CloneFromSnapshot` en el mismo espacio de nombres que la PVC que se creó originalmente. Podemos obtener todos los CloneFromSnapshots de PVC ejecutando el siguiente comando.

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

CloneFromSnapshot CRD tiene algunas fases clave para el campo `.status.phase`:

| Fase de instantánea | Descripción |
| --- | --- |
| Novedad | No se completó la clonación de la instantánea |
| InProgress | La instantánea del volumen de vSphere se está descargando desde el repositorio remoto |

| Fase de instantánea | Descripción |
| --- | --- |
| Completed | Se completó la clonación de la instantánea |
| Con errores | Error en la clonación de la instantánea |

# Realizar una copia de seguridad y restaurar cargas de trabajo de clústeres de TKG en Supervisor mediante Restic y Velero independientes

En esta sección se proporcionan temas para hacer copias de seguridad y restaurar cargas de trabajo de clústeres de TKG que se ejecutan en Supervisor mediante Velero con Restic independientes.

## Instalar y configurar Velero y Restic independientes en clústeres de TKG

Para realizar copias de seguridad y restauración de cargas de trabajo en clústeres de TKG en Supervisor, cree un almacén de datos e instale Velero con Restic en el clúster de Kubernetes.

### Descripción general

Los clústeres de TKG se ejecutan en nodos de la máquina virtual. Para realizar una copia de seguridad y restaurar cargas de trabajo de clústeres de TKG, instale Velero y Restic en el clúster.

### Requisitos previos

Asegúrese de que el entorno cumpla con los siguientes requisitos previos para instalar Velero y Restic a fin de realizar copias de seguridad y restauración de cargas de trabajo que se ejecutan en clústeres de Tanzu Kubernetes.

- Una máquina virtual Linux con suficiente almacenamiento para almacenar varias copias de seguridad de cargas de trabajo. Debe instalar MinIO en esta máquina virtual.
- Una máquina virtual Linux en la que están instaladas Herramientas de la CLI de Kubernetes para vSphere, lo que incluye complemento de vSphere para kubectl y kubectl. Debe instalar la CLI de Velero en esta máquina virtual cliente. Si no tiene una máquina virtual de este tipo, puede instalar la CLI de Velero de forma local, pero debe ajustar los pasos de instalación según corresponda.
- El entorno de Kubernetes tiene acceso a Internet y la máquina virtual cliente puede acceder a él.

## Instalar y configurar el almacén de objetos minIO

Velero requiere un almacén de objetos compatible con S3 como el destino de las copias de seguridad de la cargas de trabajo de Kubernetes. Velero admite varios proveedores de almacenes de objetos de este tipo. Por simplicidad, estas instrucciones utilizan MinIO, un servicio de almacenamiento compatible con S3 que se ejecuta localmente en la máquina virtual del almacén de objetos.

1 Instale MinIO.

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

2 Otorgue permisos de ejecución a MinIO.

```
chmod +x minio
```

3 Cree un directorio en el sistema de archivos para MinIO.

```
mkdir /DATA-MINIO
```

4 Inicie el servidor MinIO.

```
./minio server /DATA-MINIO
```

5 Una vez iniciado el servidor MinIO, se le proporcionarán detalles importantes de la instancia del almacén de datos, incluidos la URL del endpoint, AccessKey y SecretKey. Registre la URL de endpoint, AccessKey y SecretKey en la tabla.

| Metadatos del almacén de datos | Valor |
| --- | --- |
| URL de endpoint | |
| AccessKey | |
| SecretKey | |

6 Abra un navegador a la URL del endpoint del servidor MinIO y desplácese hasta el almacén de datos de MinIO.

7 Inicie sesión en el servidor MinIO y proporcione AccessKey y SecretKey.

8 Para habilitar MinIO como servicio, descargue el script `minio.service` para configurar MinIO para inicio automático.

```
curl -O https://raw.githubusercontent.com/minio/minio-service/master/linux-systemd/
minio.service
```

9 Edite el script `minio.service` y agregue el siguiente valor para `ExecStart`.

```
ExecStart=/usr/local/bin/minio server /DATA-MINIO path
```

10 Guarde el script revisado.

11 Configure el servicio MinIO mediante la ejecución de los siguientes comandos.

```
cp minio.service /etc/systemd/system
cp minio /usr/local/bin/
systemctl daemon-reload
systemctl start minio
systemctl status minio
systemctl enable minio
```

12 Cree un depósito MinIO para realizar una copia de seguridad y restauración; para ello, inicie el explorador MinIO e inicie sesión en el almacén de objetos.

13 Haga clic en el icono Crear depósito.

14 Introduzca el nombre del depósito, por ejemplo: `my-cluster-backups`.

15 Compruebe que se haya creado el depósito.

16 De forma predeterminada, un nuevo depósito MinIO es de solo lectura. Para una copia de seguridad y restauración independientes de Velero, el depósito MinIO debe ser de lectura y escritura. Para establecer el depósito en lectura y escritura, selecciónelo y haga clic en el vínculo de puntos suspensivos (puntos).

17 Seleccione **Editar directiva**.

18 Cambie la directiva a **Lectura y escritura**.

19 Haga clic en **Agregar**.

20 Para cerrar el cuadro de diálogo, haga clic en la X.

## Instalar la CLI de Velero

Instale la CLI de Velero en el cliente de máquina virtual o en la máquina local.

La versión que se utilizó para esta documentación es *Velero 1.9.7 para Tanzu Kubernetes Grid 2.2.0*.

1 Descargue Velero desde la página de descarga del producto Tanzu Kubernetes Grid en el portal de VMware Customer Connect.

**Nota** Debe utilizar el archivo binario de Velero firmado por VMware para poder recibir soporte de VMware.

2 Abra una línea de comandos y cambie el directorio a la descarga de la CLI de Velero.

3 Descomprima el archivo de descarga. Por ejemplo:

```
gunzip velero-linux-vX.X.X_vmware.1.gz
```

4 Compruebe el archivo binario de Velero.

```
ls -l
```

5 Otorgue permisos de ejecución a la CLI de Velero.

```
chmod +x velero-linux-vX.X.X_vmware.1
```

6 Haga que la CLI de Velero esté disponible globalmente, para ello, muévala a la ruta del sistema:

```
cp velero-linux-vX.X.X_vmware.1 /usr/local/bin/velero
```

7 Compruebe la instalación.

```
velero version
```

## Instalar Velero y Restic en el clúster de Tanzu Kubernetes

El contexto de la CLI de Velero seguirá automáticamente el contexto de kubectl. Antes de ejecutar los comandos de la CLI de Velero para instalar Velero y Restic en el clúster de destino, establezca el contexto de kubectl.

1 Recupere el nombre del depósito MinIO. Por ejemplo, `my-cluster-backups`.

2 Obtenga AccessKey y SecretKey para el depósito MinIO.

3 Establezca el contexto del clúster de Kubernetes de destino para que la CLI de Velero sepa en qué clúster trabajar.

```
kubectl config use-context tkgs-cluster-name
```

4 Cree un archivo de secretos denominado `credentials-minio`. Actualice el archivo con las credenciales de acceso al servidor MinIO que recopiló. Por ejemplo:

```
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

**Nota** Si recibe el mensaje de error "Error al obtener un almacén de copias de seguridad" con la descripción "NoCredentialProviders: no hay proveedores válidos en la cadena", anteponga la línea `[default]` al principio del archivo de credenciales. Por ejemplo:

```
[default]
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

5 Guarde el archivo y compruebe que esté en su lugar.

```
ls
```

6 Ejecute el siguiente comando para instalar Velero y Restic en el clúster de Kubernetes de destino. Reemplace ambas URL por la URL de la instancia de MinIO.

```
velero install \
--provider aws \
--plugins velero/velero-plugin-for-aws:v1.0.0 \
--bucket tkgs-velero \
--secret-file ./credentials-minio \
--use-volume-snapshots=false \
--use-restic \
--backup-location-config \
region=minio,s3ForcePathStyle="true",s3Url=http://10.199.17.63:9000,publicUrl=http://
10.199.17.63:9000
```

7 Compruebe la instalación de Velero y Restic.

```
kubectl logs deployment/velero -n velero
```

8 Compruebe el espacio de nombres `velero`.

```
kubectl get ns
```

9 Compruebe los pods `velero` y `restic`.

```
kubectl get all -n velero
```

## Solucionar problemas de DaemonSet de Restic (si es necesario)

Para ejecutar el DaemonSet de Restic de tres pods en un clúster de Kubernetes, es posible que deba actualizar la especificación de DaemonSet de Restic y modificar el hostPath. Para obtener más información sobre este problema, consulte Integración de Restic en la documentación de Velero.

1 Compruebe el DaemonSet de Restic de tres pods.

```
kubectl get pod -n velero
```

Si los pods tienen el estado CrashLoopBackOff, edítelos de la siguiente manera.

2 Ejecute el comando `edit`.

```
kubectl edit daemonset restic -n velero
```

3 Cambie hostPath de `/var/lib/kubelet/pods` a `/var/vcap/data/kubelet/pods`.

```
- hostPath:
    path: /var/vcap/data/kubelet/pods
```

4 Guarde el archivo.

5 Compruebe el DaemonSet de Restic de tres pods.

```
kubectl get pod -n velero
```

```
NAME                        READY   STATUS    RESTARTS   AGE
restic-5jln8                1/1     Running   0          73s
restic-bpvtq                1/1     Running   0          73s
restic-vg8j7                1/1     Running   0          73s
velero-72c84322d9-le7bd     1/1     Running   0          10m
```

## Ajustar los límites de memoria de Velero (si es necesario)

Si la copia de seguridad de Velero devuelve `status=InProgress` durante muchas horas, aumente la configuración de memoria para límites y solicitudes.

1 Ejecute el siguiente comando.

```
kubectl edit deployment/velero -n velero
```

2 Cambie la configuración de memoria para límites y solicitudes desde el valor predeterminado de `256Mi` y `128Mi` a `512Mi` y `256Mi`.

```
ports:
- containerPort: 8085
  name: metrics
  protocol: TCP
resources:
  limits:
    cpu: "1"
    memory: 512Mi
  requests:
    cpu: 500m
    memory: 256Mi
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
```

# Copia de seguridad y restauración de cargas de trabajo de clúster mediante Restic y Velero independientes

Puede realizar operaciones de copia de seguridad y restauración de las cargas de trabajo que se ejecutan en los clústeres de TKG mediante Velero y Restic independientes. Este método es una alternativa al uso de un complemento complemento de Velero para vSphere. La razón principal para usar Velero independiente es la necesidad de portabilidad. Se requiere Restic para las cargas de trabajo con estado.

## Requisitos previos

Para realizar copias de seguridad y restaurar cargas de trabajo en clústeres de TKG mediante Restic y Velero independientes, debe instalar la versión independiente de estos en el clúster de destino. Si la restauración se va a realizar en un clúster de destino independiente, Velero y Restic también deben estar instalados en el clúster de destino. Consulte Instalar y configurar Velero y Restic independientes en clústeres de TKG.

## Realizar una copia de seguridad de una aplicación sin estado que se ejecuta en un clúster de TKG

La copia de seguridad de una aplicación sin estado que se ejecuta en un clúster de TKG requiere el uso de Velero.

En este ejemplo se muestra cómo realizar una copia de seguridad y restaurar una aplicación sin estado de ejemplo mediante la etiqueta `--include namespaces` en la que todos los componentes de la aplicación se encuentran en ese espacio de nombres.

```
velero backup create example-backup --include-namespaces example-backup
```

Debería ver el siguiente mensaje:

```
Backup request "example-backup" submitted successfully.
Run `velero backup describe example-backup` or `velero backup logs example-backup` for more
details.
```

Compruebe la copia de seguridad que se creó.

```
velero backup get
```

```
velero backup describe example-backup
```

Compruebe el depósito Velero en el almacén de objetos compatible con S3, como el servidor MinIO.

Velero escribe algunos metadatos en definiciones de recursos personalizados (Custom Resource Definitions, CRD) de Kubernetes.

```
kubectl get crd
```

Las CRD de Velero permiten ejecutar ciertos comandos, como los siguientes:

```
kubectl get backups.velero.io -n velero
```

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

## Restaurar una aplicación sin estado en ejecución en un clúster de TKG

La restauración de una aplicación sin estado que se ejecuta en un clúster de TKG requiere el uso de Velero.

Para probar la restauración de una aplicación de ejemplo, elimínela.

Elimine el espacio de nombres:

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

Restaure la aplicación:

```
velero restore create --from-backup example-backup
```

Debería ver el siguiente mensaje:

```
Restore request "example-backup-20200721145620" submitted successfully.
Run `velero restore describe example-backup-20200721145620` or `velero restore logs example-
backup-20200721145620` for more details.
```

Compruebe que la aplicación se restauró:

```
velero restore describe example-backup-20200721145620
```

Ejecute los siguientes comandos para comprobar:

```
velero restore get
```

```
kubectl get ns
```

```
kubectl get pod -n example
```

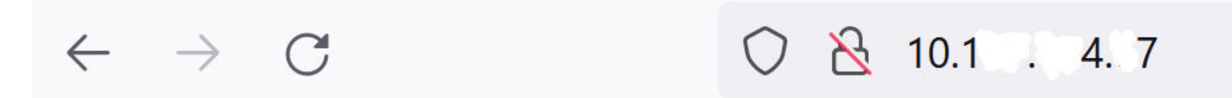
```
kubectl get svc -n example
```

## Realizar una copia de seguridad de una aplicación con estado que se ejecuta en un clúster de TKG

Realizar una copia de seguridad de una aplicación con estado que se ejecuta en un clúster de TKG implica realizar una copia de seguridad de los metadatos de la aplicación y los datos de la aplicación almacenados en un volumen persistente. Para ello, necesita Velero y Restic.

En este ejemplo, utilizaremos la aplicación del libro de visitas. Si se supone que implementó la aplicación del libro de visitas en un clúster de TKG. Consulte Implementar la aplicación del libro de visitas en un clúster de TKG.

Para que podamos demostrar una copia de seguridad y una restauración con estado, envíe un mensaje a la aplicación del libro de visitas mediante la página web de front-end para que los mensajes persistan. Por ejemplo:




Guestbook

Messages

Submit

message 1
message 2
message 3

En este ejemplo se muestra cómo realizar una copia de seguridad y restaurar la aplicación del libro de visitas mediante la etiqueta `--include namespace`, así como las anotaciones del pod.

**Nota** En este ejemplo, se utilizan anotaciones. Sin embargo, ya no se necesitan anotaciones para Velero 1.5 y versiones posteriores. Para no utilizar anotaciones, puede usar la opción `--default-volumes-to-restic` al crear la copia de seguridad. Esto hará una copia de seguridad automática de todos los VA mediante Restic. Consulte https://velero.io/docs/v1.5/restic/ para obtener más información.

Para comenzar el procedimiento de copia de seguridad, obtenga los nombres de los pods:

```
kubectl get pod -n guestbook
```

Por ejemplo:

```
kubectl get pod -n guestbook

NAME                                            READY   STATUS    RESTARTS   AGE
guestbook-frontend-deployment-85595f5bf9-h8cff  1/1     Running          0          55m
guestbook-frontend-deployment-85595f5bf9-lw6tg  1/1     Running          0          55m
guestbook-frontend-deployment-85595f5bf9-wpqc8  1/1     Running          0          55m
redis-leader-deployment-64fb8775bf-kbs6s        1/1     Running          0          55m
redis-follower-deployment-84cd76b975-jrn8v      1/1     Running          0          55m
redis-follower-deployment-69df9b5688-zml4f      1/1     Running          0          55m
```

Los volúmenes persistentes se asocian a los pods de Redis. Debido a que estamos realizando una copia de seguridad de estos pods con estado con Restic, es necesario agregar anotaciones a los pods con estado con el nombre `volumeMount`.

Debe conocer `volumeMount` para anotar el pod con estado. Para obtener `mountName`, ejecute el siguiente comando.

```
kubectl describe pod redis-leader-deployment-64fb8775bf-kbs6s -n guestbook
```

En los resultados, verá `Containers.leader.Mounts: /data` de `redis-leader-data`. Este último token es el nombre `volumeMount` que se utilizará para la anotación del pod principal. Para el seguidor, será `redis-follower-data`. También puede obtener el nombre de `volumeMount` del YAML de origen.

Anote cada uno de los pods de Redis, por ejemplo:

```
kubectl -n guestbook annotate pod redis-leader-64fb8775bf-kbs6s backup.velero.io/backup-
volumes=redis-leader-data
```

Debería ver el siguiente mensaje:

```
pod/redis-leader-64fb8775bf-kbs6s annotated
```

Verifique las anotaciones:

```
kubectl -n guestbook describe pod redis-leader-64fb8775bf-kbs6s | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-leader-data
```

```
kubectl -n guestbook describe pod redis-follower-779b6d8f79-5dphr | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-follower-data
```

Realice la copia de seguridad de Velero:

```
velero backup create guestbook-backup --include-namespaces guestbook
```

Debería ver el siguiente mensaje:

```
Backup request "guestbook-backup" submitted successfully.
Run `velero backup describe guestbook-pv-backup` or `velero backup logs guestbook-pv-backup`
for more details.
```

Compruebe la copia de seguridad que se creó.

```
velero backup get

NAME                  STATUS      ERRORS   WARNINGS   CREATED
EXPIRES   STORAGE LOCATION   SELECTOR
guestbook-backup      Completed   0        0          2020-07-23 16:13:46 -0700 PDT
29d       default            <none>
```

Compruebe los detalles de la copia de seguridad.

```
velero backup describe guestbook-backup --details
```

Tenga en cuenta que Velero permite ejecutar otros comandos, como:

```
kubectl get backups.velero.io -n velero

NAME               AGE
guestbook-backup   4m58s
```

Y:

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

## Restaurar una aplicación con estado que se ejecuta en un clúster de TKG 2.0

La restauración de una aplicación con estado que se ejecuta en un clúster de TKG implica restaurar tanto los metadatos de la aplicación como los datos de la aplicación almacenados en un volumen persistente. Para ello, necesita Velero y Restic.

En este ejemplo, se supone que se realizó una copia de seguridad de la aplicación de libro de visitas con estado, como se describe en la sección anterior.

Para probar la restauración de la aplicación con estado, elimine su espacio de nombres:

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

Verifique la eliminación de la aplicación:

```
kubectl get ns
kubectl get pvc,pv --all-namespaces
```

Para restaurar una aplicación desde una copia de seguridad, utilice la siguiente sintaxis de comandos.

```
velero restore create --from-backup <velero-backup-name>
```

Por ejemplo:

```
velero restore create --from-backup guestbook-backup
```

Deben aparecer mensajes similares al siguiente:

```
Restore request "guestbook-backup-20200723161841" submitted successfully.
Run `velero restore describe guestbook-backup-20200723161841` or `velero restore logs
guestbook-backup-20200723161841` for more details.
```

Compruebe que se restauró la aplicación del libro de visitas con estado:

```
velero restore describe guestbook-backup-20200723161841

Name:         guestbook-backup-20200723161841
Namespace:    velero
Labels:       <none>
Annotations:  <none>

Phase:  Completed

Backup:  guestbook-backup

Namespaces:
  Included:  all namespaces found in the backup
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        nodes, events, events.events.k8s.io, backups.velero.io,
restores.velero.io, resticrepositories.velero.io
  Cluster-scoped:  auto

Namespace mappings:  <none>

Label selector:  <none>

Restore PVs:  auto

Restic Restores (specify --details for more information):
  Completed:  3
```

Ejecute el siguiente comando adicional para verificar la restauración:

```
velero restore get

NAME                                  BACKUP                STATUS      ERRORS   WARNINGS
CREATED                          SELECTOR
guestbook-backup-20200723161841      guestbook-backup      Completed   0        0
2021-08-11 16:18:41 -0700 PDT   <none>
```

Compruebe que se restauró el espacio de nombres:

```
kubectl get ns

NAME               STATUS   AGE
default            Active   16d
guestbook          Active   76s
...
velero             Active   2d2h
```

Compruebe que la aplicación se restauró:

```
vkubectl get all -n guestbook

NAME                                READY   STATUS    RESTARTS   AGE
pod/frontend-6cb7f8bd65-h2pnb       1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-kwlpr       1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-snwl4       1/1     Running   0          6m27s
pod/redis-leader-64fb8775bf-kbs6s   1/1     Running   0          6m28s
pod/redis-follower-779b6d8f79-5dphr 1/1     Running   0          6m28s
pod/redis-follower-899c7e2z65-8apnk 1/1     Running   0          6m28s

NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend           LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower               ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                 ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

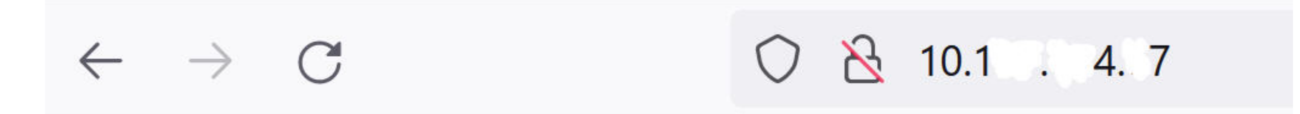
Compruebe que se restauren los volúmenes persistentes:

```
kubectl get pvc,pv -n guestbook

NAME                                       STATUS
VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/redis-leader-claim   Bound    pvc-a2f6e6d4-42db-4fb8-
a198-5379a2552509   2Gi        RWO            thin-disk      2m40s
persistentvolumeclaim/redis-follower-claim Bound    pvc-55591938-921f-452a-
b418-2cc680c0560b   2Gi        RWO            thin-disk      2m40s

NAME                                                        CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS   CLAIM                             STORAGECLASS   REASON   AGE
persistentvolume/pvc-55591938-921f-452a-b418-2cc680c0560b   2Gi        RWO
Delete           Bound    guestbook/redis-follower-claim    thin-disk               2m40s
persistentvolume/pvc-a2f6e6d4-42db-4fb8-a198-5379a2552509   2Gi        RWO
Delete           Bound    guestbook/redis-leader-claim      thin-disk               2m40s
```

Por último, acceda al front-end del libro de visitas mediante la dirección IP externa del servicio de front-end del libro de visitas y compruebe que se restauren los mensajes que envió al principio del tutorial. Por ejemplo:




## Copia de seguridad y restauración mediante Velero con instantánea de CSI

Puede utilizar Velero con instantánea de CSI para realizar una copia de seguridad y una restauración de los volúmenes persistentes creados por CSI para cargas de trabajo que se ejecutan en clústeres de TKG aprovisionados en Supervisor.

### Requisitos

Cumpla con los siguientes requisitos:

- vSphere 8.0 U2 o versiones posteriores
- Tanzu Kubernetes 1.26.5 para vSphere 8.x o versiones posteriores
- Volúmenes persistentes creados con controladores de CSI compatibles con instantáneas de volumen

**Atención** El uso de Velero con instantánea de CSI solo está disponible para volúmenes persistentes creados con controladores de CSI que admiten instantáneas de volumen. Consulte Crear instantáneas en un clúster de TKG en *Uso del servicio TKG con el plano de control de IaaS de vSphere* para obtener más información.

## Procedimiento

Puede utilizar Velero con una instantánea de interfaz de almacenamiento de contenedores (Container Storage Interface, CSI) para realizar copias de seguridad y restauración de cargas de trabajo que se ejecutan en clústeres de TKGS. El agente del nodo de Velero es un DaemonSet que aloja módulos para completar tareas concretas de copia de seguridad y restauración mediante el movimiento de datos de instantáneas de CSI. Para obtener más información, consulte Compatibilidad con instantáneas de interfaz de almacenamiento de contenedores en Velero.

1 Cree una ubicación de almacenamiento compatible con S3, como MinIO o un contenedor AWS S3.

   En el siguiente ejemplo, se utiliza un contenedor AWS S3.

   Para utilizar MinIO, consulte Instalar y configurar el almacén de objetos minIO.

2 Instale la CLI de Velero en el cliente de clúster en el que está ejecutando kubectl.

   Descárguela desde https://github.com/vmware-tanzu/velero/releases.

   Consulte las instrucciones de instalación en uno de los siguientes vínculos:

   - Paso 1: Instalar la CLI de Velero en una Workstation de Linux
   - Instalar la CLI de Velero
   - https://velero.io/docs/v1.12/basic-install/#install-the-cli

3 Conéctese al clúster de Servicio TKG en el que desea ejecutar la copia de seguridad de Velero.

   Consulte Conectarse a un clúster de Servicio TKG como usuario de vCenter Single Sign-On con Kubectl.

4 Ejecute el comando de instalación de Velero, por ejemplo, con un almacenamiento de AWS S3 y el archivo de credenciales correspondiente.

```
velero install \
    --provider aws \
    --plugins velero/velero-plugin-for-aws:v1.9.0,velero/velero-plugin-for-csi:v0.7.0 \
    --bucket velero-cpe-backup-bucket \
    --secret-file ./cloud-credential \
    --use-volume-snapshots=true \
    --features=EnableCSI --use-node-agent
```

   **Nota** A partir de la versión 1.14 de Velero, el complemento CSI de Velero se combina con Velero. Por lo tanto, si va a instalar Velero v1.14 o una versión posterior, no es necesario instalar el complemento CSI de Velero. Si lo hace, el pod de Velero no se inicia.

# Solucionar problemas de clústeres de servicio TKG

21

Puede solucionar los problemas de los clústeres de servicio TKG con una variedad de métodos, como conectarse a cualquier nodo de clúster, ver la jerarquía de recursos del clúster y recopilar archivos de registro.

Lea los siguientes temas a continuación:

- Extraer registros para solucionar problemas de clústeres de TKG en Supervisor
- Comprobar el estado de los componentes de TKG en Supervisor
- Solucionar problemas de conexión del clúster de TKG y errores de inicio de sesión
- Solucionar errores de la biblioteca de contenido
- Corregir errores de clase de máquina virtual
- Solucionar errores de aprovisionamiento de clústeres de TKGS
- Solucionar errores de nodo del clúster de servicio TKG
- Solucionar errores de redes del clúster de servicio TKG
- Reiniciar una actualización de clúster de TKG con errores
- Corregir errores de implementación de contenedores
- Solucionar errores del registro del contenedor
- Solucionar errores con certificados de CA de confianza adicionales

## Extraer registros para solucionar problemas de clústeres de TKG en Supervisor

Consulte este tema para extraer varios registros para solucionar problemas de clústeres de TKG en Supervisor, incluidos un paquete de soporte de Supervisor, el registro de administración de cargas de trabajo y los registros CAPI, CAPV, operador de máquina virtual y administrador de controladoras de TKG.

## Recopilar un paquete de soporte para Supervisor

Para solucionar errores en los clústeres de TKG, puede exportar los registros del Supervisor. Por lo general, la revisión de estos registros se realiza con asesoramiento del soporte de VMware.

1 Inicie sesión en su entorno de vSphere IaaS control plane mediante vSphere Client.

2 Seleccione **Menú > Administración de cargas de trabajo**.

3 Seleccione la pestaña **Supervisor**.

4 Seleccione la instancia de **Supervisor** de destino.

5 Seleccione **Exportar registros**.

Una vez que haya recopilado el paquete de soporte, consulte el siguiente artículo de la base de conocimientos: Cómo cargar información de diagnóstico para VMware a través del portal de FTP seguro: http://kb.vmware.com/kb/2069559. Consulte también Aprovisionamiento de registros para vSpehre with Tanzu.

## Recopilar paquete de soporte para un clúster de TKG

Puede utilizar la utilidad TKC Support Bundler para recopilar archivos de registro del clúster de TKG y solucionar problemas.

Para obtener y utilizar la utilidad TKC Support Bundler, consulte el artículo Aprovisionamiento de registros para vSpehre with Tanzu en la base de conocimientos de soporte de VMware.

## Poner en cola el archivo de registro de administración de cargas de trabajo

Poner en cola el archivo de registro del plano de control de carga de trabajo (Workload Control Plane, WCP) puede ayudar a solucionar errores en los clústeres de TKG y el Supervisor.

1 Establezca una conexión SSH con vCenter Server Appliance.

2 Inicie sesión como usuario `root`.

3 Ejecute el comando `shell`.

   Verá lo siguiente:

```
Shell access is granted to root
root@localhost [ ~ ]#
```

4 Ejecute el siguiente comando para seguir el archivo de registro de WCP.

```
tail -f /var/log/vmware/wcp/wcpsvc.log
```

# Recopilar registros específicos de TKG en Supervisor

Supervisor ejecuta varios pods de Kubernetes que proporcionan infraestructura a TKG 2.0.

```
kubectl -n vmware-system-capw get deployments.apps
NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
capi-controller-manager                         2/2     2            2           18h
capi-kubeadm-bootstrap-controller-manager       2/2     2            2           18h
capi-kubeadm-control-plane-controller-manager   2/2     2            2           18h
capv-controller-manager                         2/2     2            2           10h
capw-controller-manager                         2/2     2            2           18h
capw-webhook                                    2/2     2            2           18h
```

Los pods de infraestructura son implementaciones que ejecutan réplicas. Es posible que deba identificar qué réplica es el líder y revisar sus registros para ver la última versión. Por lo general, un componente que no es líder se detiene después de registrar algo sobre un intento de adquirir la concesión.

Tendrá que iniciar sesión en Supervisor y utilizar el espacio de nombres de vSphere de Supervisor para comprobar estos pods.

Los registros que utilizan un selector de etiquetas no funcionan bien, por lo que es posible que deba descartar la cadena aleatoria que se agrega al final del nombre del pod. A veces, canalizar el resultado a grep error o grep -i error es un inicio útil. Por ejemplo, `kubectl logs <args> | grep error`.

### Registros de CAPI

Proveedor de API de clúster:

```
kubectl logs -n vmware-system-capw -c manager vmware-system-capw-capi-controller-manager-
<id>
```

### Registros de CAPV

API de clúster para proveedor de vSphere:

```
kubectl logs -n vmware-system-capv -c manager vmware-system-capw-v1alpha3-vmware-system-
capv-v1alpha3-controller-manager-<id>
```

### Registros del operador de máquina virtual

Operador de máquina virtual:

```
kubectl logs -n vmware-system-vmop -c manager vmware-system-vmop-controller-manager-<id>
```

### Registros del administrador de controladoras de TKG

Administrador de controladores de GCM

```
kubectl logs -n vmware-system-tkg -c manager vmware-system-tkg-controller-manager-<id>
```

# Comprobar el estado de los componentes de TKG en Supervisor

Consulte este tema para conocer varias técnicas que se utilizan para comprobar el estado de Supervisor con respecto a los componentes de TKG.

## Comprobar el estado de los pods de Supervisor

Los pods de Supervisor ejecutan componentes de infraestructura de TKG.

Compruebe si todos los pods de Supervisor están en un estado de ejecución.

```
kubectl get pods -A | grep "Running"
```

**Nota** Otra opción es utilizar `grep -v "Running"` para devolver los pods que no están en ejecución.

Por ejemplo:

```

NAMESPACE
NAME                                                                  READY   STATUS
RESTARTS        AGE
kube-system
coredns-855c5b4cfd-8w4hp                                              1/1     Running
0               27d
kube-system                                 coredns-855c5b4cfd-
bx2hk                                          1/1     Running     0               27d
kube-system                                 coredns-855c5b4cfd-
rrb5n                                          1/1     Running     0               27d
kube-system                                 docker-
registry-423f01b9b30c727e9c237a0031999b14                 1/1     Running
0               27d
kube-system                                 docker-
registry-423f568f75dcb48725b0d768b7e4bdf5                 1/1     Running
0               27d
kube-system                                 docker-
registry-423f930ca2413d96beef34526c2e61b4                 1/1     Running
0               27d
kube-system
etcd-423f01b9b30c727e9c237a0031999b14                                 1/1     Running     1 (27d
ago)     27d
kube-system
etcd-423f568f75dcb48725b0d768b7e4bdf5                                 1/1     Running     1 (27d
ago)     27d
kube-system
etcd-423f930ca2413d96beef34526c2e61b4                                 1/1     Running     1 (27d
ago)     27d
kube-system                                 kube-
apiserver-423f01b9b30c727e9c237a0031999b14                  1/1     Running     1 (27d
ago)     27d
kube-system                                 kube-
```

```
apiserver-423f568f75dcb48725b0d768b7e4bdf5                   1/1     Running     1 (27d
ago)     27d
kube-system                                 kube-
apiserver-423f930ca2413d96beef34526c2e61b4                   1/1     Running     1 (27d
ago)     27d
kube-system                                 kube-controller-
manager-423f01b9b30c727e9c237a0031999b14          1/1     Running     0               27d
kube-system                                 kube-controller-
manager-423f568f75dcb48725b0d768b7e4bdf5          1/1     Running     0               27d
kube-system                                 kube-controller-
manager-423f930ca2413d96beef34526c2e61b4          1/1     Running     0               27d
kube-system                                 kube-
proxy-8h499                                                  1/1     Running
0               27d
kube-system                                 kube-proxy-
bm7qt                                             1/1     Running     0               27d
kube-system                                 kube-proxy-
dnmq2                                             1/1     Running     0               27d
kube-system                                 kube-
scheduler-423f01b9b30c727e9c237a0031999b14                   2/2     Running     13 (25d
ago)    27d
kube-system                                 kube-
scheduler-423f568f75dcb48725b0d768b7e4bdf5                   2/2     Running
0               27d
kube-system                                 kube-
scheduler-423f930ca2413d96beef34526c2e61b4                   2/2     Running
0               27d
kube-system                                 kubectl-plugin-
vsphere-423f01b9b30c727e9c237a0031999b14           1/1     Running     3 (27d ago)     27d
kube-system                                 kubectl-plugin-
vsphere-423f568f75dcb48725b0d768b7e4bdf5           1/1     Running     3 (27d ago)     27d
kube-system                                 kubectl-plugin-
vsphere-423f930ca2413d96beef34526c2e61b4           1/1     Running     3 (27d ago)     27d
kube-system                                 wcp-
authproxy-423f01b9b30c727e9c237a0031999b14                    1/1     Running
0               27d
kube-system                                 wcp-
authproxy-423f568f75dcb48725b0d768b7e4bdf5                    1/1     Running
0               27d
kube-system                                 wcp-
authproxy-423f930ca2413d96beef34526c2e61b4                    1/1     Running
0               27d
kube-system                                 wcp-
fip-423f01b9b30c727e9c237a0031999b14                          1/1     Running
0               27d
kube-system                                 wcp-
fip-423f568f75dcb48725b0d768b7e4bdf5                          1/1     Running
0               27d
kube-system                                 wcp-
fip-423f930ca2413d96beef34526c2e61b4                          1/1     Running
0               27d
svc-tmc-c63                                 agent-updater-69f6598bcd-
zrkwq                                    1/1     Running     0               27d
svc-tmc-c63                                 agentupdater-workload-27696934--1-
vz5sg                            0/1     Completed   0               35s
```

```
svc-tmc-c63                                 cluster-health-
extension-68948f657-4gpcd                          1/1     Running     0               27d
svc-tmc-c63                                 extension-manager-f8886bfb7-
vdsm9                                 1/1     Running     0               27d
svc-tmc-c63                                 extension-updater-79b4787cf6-
bwssn                                1/1     Running     0               27d
svc-tmc-c63                                 intent-agent-66576db5bd-
lj2gk                                     1/1     Running     0               5d6h
svc-tmc-c63                                 sync-agent-
f9c68cc58-6zddj                                       1/1     Running     0               6d
svc-tmc-c63                                 tmc-agent-installer-27696934--1-
jgwvw                               0/1     Completed   0               35s
svc-tmc-c63                                 tmc-auto-attach-6488b9cd8b-
xdfzz                                  1/1     Running     0               18h
svc-tmc-c63                                 vsphere-resource-
retriever-58985c99cb-68h6v                          1/1     Running     0               18h
vmware-system-appplatform-operator-system   vmware-system-appplatform-operator-
mgr-0                            1/1     Running     0               27d
vmware-system-appplatform-operator-system   vmware-system-psp-operator-mgr-587f66646d-
xxvmr                     1/1     Running     0               27d
vmware-system-capw                          capi-controller-
manager-766c6fc449-4qqvf                          2/2     Running     423 (26d ago)   27d
vmware-system-capw                          capi-controller-manager-766c6fc449-
bcpdq                            2/2     Running     410 (26d ago)   27d
vmware-system-capw                          capi-controller-manager-766c6fc449-
rnznx                            2/2     Running     0               26d
vmware-system-capw                          capi-kubeadm-bootstrap-controller-
manager-58fd767b49-585f2        2/2     Running     402 (25d ago)   27d
vmware-system-capw                          capi-kubeadm-bootstrap-controller-
manager-58fd767b49-96q6m        2/2     Running     398 (25d ago)   27d
vmware-system-capw                          capi-kubeadm-bootstrap-controller-
manager-58fd767b49-nssgq        2/2     Running     407 (25d ago)   27d
vmware-system-capw                          capi-kubeadm-control-plane-controller-
manager-559df997b-762jr     2/2     Running     193 (26d ago)   27d
vmware-system-capw                          capi-kubeadm-control-plane-controller-
manager-559df997b-bb42s     2/2     Running     189 (26d ago)   27d
vmware-system-capw                          capi-kubeadm-control-plane-controller-
manager-559df997b-wxhqv     2/2     Running     199 (26d ago)   27d
vmware-system-capw                          capw-controller-
manager-6dd47d75b-6ncxk                          2/2     Running     400 (25d ago)   27d
vmware-system-capw                          capw-controller-manager-6dd47d75b-
k2ph4                           2/2     Running     399 (25d ago)   27d
vmware-system-capw                          capw-controller-manager-6dd47d75b-
np9sg                           2/2     Running     403 (25d ago)   27d
vmware-system-capw                          capw-
webhook-5484757c7-2pkbt                                     2/2     Running
0               27d
vmware-system-capw                          capw-webhook-5484757c7-
fkt7z                                    2/2     Running     0               27d
vmware-system-capw                          capw-webhook-5484757c7-
r85kw                                    2/2     Running     0               27d
vmware-system-cert-manager                  cert-manager-6ccbcfcd57-
lppgn                                   1/1     Running     1 (27d ago)     27d
vmware-system-cert-manager                  cert-manager-
cainjector-796f7b74db-5qvgn                          1/1     Running     3 (27d ago)     27d
```

```
vmware-system-cert-manager                cert-manager-webhook-586948846f-
b584m                            1/1     Running     0               27d
vmware-system-csi                         vsphere-csi-
controller-6d8cfd75cd-66zbj                        6/6     Running     0               27d
vmware-system-csi                         vsphere-csi-controller-6d8cfd75cd-
b4nhz                          6/6     Running     1 (27d ago)     27d
vmware-system-csi                         vsphere-csi-controller-6d8cfd75cd-
v6hlf                          6/6     Running     0               27d
vmware-system-kubeimage                   image-controller-ff79fb5fc-
kd6ts                                  1/1     Running     0               27d
vmware-system-license-operator            vmware-system-license-operator-controller-
manager-7d555768bnxjb   1/1     Running     0               25d
vmware-system-license-operator            vmware-system-license-operator-controller-
manager-7d555768j2sb8   1/1     Running     0               25d
vmware-system-license-operator            vmware-system-license-operator-controller-
manager-7d555768w7v77   1/1     Running     0               25d
vmware-system-logging                     fluentbit-
p24gk                                               1/1     Running     0
27d
vmware-system-logging                     fluentbit-
rj2t8                                               1/1     Running     0
27d
vmware-system-logging                     fluentbit-
xx2lk                                               1/1     Running     0
27d
vmware-system-nsop                        vmware-system-nsop-controller-
manager-65b8445959-66msw            1/1     Running     0               27d
vmware-system-nsop                        vmware-system-nsop-controller-manager-65b8445959-
nm6xh            1/1     Running     0               27d
vmware-system-nsop                        vmware-system-nsop-controller-manager-65b8445959-
sv5w7            1/1     Running     0               27d
vmware-system-nsx                         nsx-ncp-6f989c9c67-
vb4x6                                        1/1     Running     5 (27d ago)     27d
vmware-system-registry                    vmware-registry-controller-
manager-7f49485b9-72kh7                2/2     Running     0               27d
vmware-system-tkg                         masterproxy-tkgs-
plugin-8npzx                                  1/1     Running     0               27d
vmware-system-tkg                         masterproxy-tkgs-plugin-
bjtsz                                    1/1     Running     0               27d
vmware-system-tkg                         masterproxy-tkgs-plugin-
v92gt                                    1/1     Running     0               27d
vmware-system-tkg                         tkgs-plugin-server-5fc4c985c7-
bz8jh                               1/1     Running     0               27d
vmware-system-tkg                         tkgs-plugin-server-5fc4c985c7-
r9wj5                               1/1     Running     0               27d
vmware-system-tkg                         tkgs-plugin-server-5fc4c985c7-
sdr55                               1/1     Running     0               27d
vmware-system-tkg                         vmware-system-tkg-controller-manager-7ffcc55df5-
dqkkm             2/2     Running     0               25d
vmware-system-tkg                         vmware-system-tkg-controller-manager-7ffcc55df5-
hkvx9             2/2     Running     0               25d
vmware-system-tkg                         vmware-system-tkg-controller-manager-7ffcc55df5-
txxrf             2/2     Running     0               25d
vmware-system-tkg                         vmware-system-tkg-state-
metrics-5bbb6d668c-7c5vt                 2/2     Running     238 (26d ago)   27d
```

```
vmware-system-tkg                          vmware-system-tkg-state-metrics-5bbb6d668c-
c87zs                  2/2     Running     237 (26d ago)   27d
vmware-system-tkg                          vmware-system-tkg-state-metrics-5bbb6d668c-
wc46p                  2/2     Running     237 (26d ago)   27d
vmware-system-tkg                          vmware-system-tkg-
webhook-567f9fd68c-425xs                       2/2     Running     0               25d
vmware-system-tkg                          vmware-system-tkg-
webhook-567f9fd68c-97d6z                       2/2     Running     0               25d
vmware-system-tkg                          vmware-system-tkg-webhook-567f9fd68c-
dnkgt                        2/2     Running     0               25d
vmware-system-ucs                          upgrade-compatibility-service-5745846d58-
tpk67                    1/1     Running     0               27d
vmware-system-ucs                          upgrade-compatibility-service-5745846d58-
twxkt                    1/1     Running     0               27d
vmware-system-ucs                          upgrade-compatibility-service-5745846d58-
wzl8x                    1/1     Running     0               27d
vmware-system-vmop                         vmware-system-vmop-controller-manager-
c8499b9df-5h6f9             2/2     Running     0               27d
vmware-system-vmop                         vmware-system-vmop-controller-manager-
c8499b9df-6wgr7             2/2     Running     0               27d
vmware-system-vmop                         vmware-system-vmop-controller-manager-c8499b9df-
tvbg6             2/2     Running     0               27d
vmware-system-vmop                         vmware-system-vmop-hostvalidator-8498cc5f4d-
vqhnk                 1/1     Running     0               27d
```

Si algún pod de Supervisor no está en estado de ejecución, revise ese pod con el siguiente comando.

```
kubectl describe pod <POD Name> -n <Namespace>
```

## Comprobar el estado de los recursos de Supervisor

Recursos de controladora TKG:

```
kubectl get tkc
```

Recursos de API del clúster (CAPI, CABPK, CAPW, CAPV):

```
kubectl get cluster-api
```

Recursos del operador de máquina virtual:

```
kubectl get virtualmachines,virtualmachineservices,virtualmachinesetresourcepolicies
```

Recurso del operador de máquina virtual, clúster en su ámbito y sincronizado desde la biblioteca de contenido:

```
kubectl get virtualmachineimages
```

Recursos de almacenamiento:

```
kubectl get persistentvolumeclaims,cnsnodevmattachment,cnsvolumemetadatas
```

Recursos de redes (específicos de NSX):

```
kubectl get service,lb,lbm,vnet,vnetif,nsxerrors,nsxnetworkinterfaces
```

Obtenga todos los recursos de Supervisor y escríbalos en un archivo:

```
kubectl api-resources --namespaced -o name | paste -d',' -s | xargs kubectl get -n
<namespace> > resources_in_namespace.txt
```

## Comprobar que haya implementaciones de API del clúster

Compruebe que haya implementaciones de CAPI, CAPW, CAPV.

```
kubectl -n vmware-system-capw get deployments.apps
NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
capi-controller-manager                       2/2     2            2           18h
capi-kubeadm-bootstrap-controller-manager     2/2     2            2           18h
capi-kubeadm-control-plane-controller-manager   2/2     2            2           18h
capv-controller-manager                       2/2     2            2           10h
capw-controller-manager                       2/2     2            2           18h
capw-webhook                                  2/2     2            2           18h
```

## Comprobar los archivos de paquetes de soporte

La carpeta `commands/` del Recopilar un paquete de soporte para Supervisor tiene registros journalctl que proporcionan detalles sobre lo que ocurre durante el proceso de encendido de WCP.

`kubectl_describe_virtualmachine.txt`

`kubectl_describe_tanzukubernetescluster.txt`

`kubectl_describe_kubeadmconfig.txt`

`kubectl-describe-pod_kube-system.txt`

`kubectl-describe-pod_vmware-system-capw.txt`

`kubectl-describe-pod_vmware-system-tkg.txt`

`kubectl-describe-pod_vmware-system-ucs.txt`

`kubectl-describe-pod_vmware-system-vmop.txt`

`kubectl_describe_cluster_resource_virtualmachineimages.txt`

`docker_images.txt`

## Comprobar el estado de un clúster de TKG

Compruebe que todos los nodos del clúster (máquinas virtuales) estén en un estado listo.

```
kubectl get nodes -o wide
```

```
NAME                                                          STATUS   ROLES
AGE   VERSION             INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                 KERNEL-
VERSION   CONTAINER-RUNTIME
tkgs-cluster-13-control-plane-dpmjj                           Ready    control-plane,master
12d   v1.22.9+vmware.1   10.244.0.25   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-control-plane-nb5r6                           Ready    control-plane,master
12d   v1.22.9+vmware.1   10.244.0.18   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-control-plane-zpcgs                           Ready    control-plane,master
12d   v1.22.9+vmware.1   10.244.0.26   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a1-gq458-9d6458d6f-c7t8c    Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.24   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a1-gq458-9d6458d6f-slzvn    Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.19   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a1-gq458-9d6458d6f-vzrsd    Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.22   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a2-tw99z-7b547b7f85-k5h4s   Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.20   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a2-tw99z-7b547b7f85-lkmdx   Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.21   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
tkgs-cluster-13-worker-nodepool-a2-tw99z-7b547b7f85-qwv98   Ready    <none>
12d   v1.22.9+vmware.1   10.244.0.23   <none>        VMware Photon OS/Linux
4.19.225-3.ph3   containerd://1.5.11
```

Compruebe que todos los pods estén en ejecución.

```
kubectl get pods -A
```

```
NAMESPACE
NAME                                                                            READY
STATUS      RESTARTS      AGE
kube-system                    antrea-
agent-58hv7                                                              2/2     Running
0             12d
kube-system                    antrea-
agent-6x897                                                              2/2     Running
0             12d
kube-system                    antrea-
agent-7d99k                                                              2/2     Running
0             12d
kube-system                    antrea-agent-
```

```
b7vdv                                                      2/2     Running
0            12d
kube-system                antrea-agent-
dhdlg                                                      2/2     Running
0            12d
kube-system                antrea-agent-
mj4wx                                                      2/2     Running
0            12d
kube-system                antrea-agent-
v7vtv                                                      2/2     Running
0            12d
kube-system                antrea-agent-
x49gz                                                      2/2     Running     1 (12d
ago)   12d
kube-system                antrea-agent-
z2gth                                                      2/2     Running
0            12d
kube-system                antrea-controller-bb59f5fbf-
t6cm9                                     1/1     Running     0            12d
kube-system                antrea-resource-
init-65b586c9db-2cbxx                                1/1     Running     0
12d
kube-system
coredns-5f64c4fff8-2gsqn                                            1/1
Running     0            12d
kube-system                coredns-5f64c4fff8-
hvkg9                                          1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-control-plane-
dpmjj                     1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-control-plane-
nb5r6                     1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-control-plane-
zpcgs                     1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a1-
gq458-9d6458d6f-c7t8c    1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a1-
gq458-9d6458d6f-slzvn    1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a1-
gq458-9d6458d6f-vzrsd    1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a2-
tw99z-7b547b7f85-k5h4s   1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a2-
tw99z-7b547b7f85-lkmdx   1/1     Running     0            12d
kube-system                docker-registry-tkgs-cluster-13-worker-nodepool-a2-
tw99z-7b547b7f85-qwv98   1/1     Running     0            12d
kube-system                etcd-tkgs-cluster-13-control-plane-
dpmjj                              1/1     Running     0            12d
kube-system                etcd-tkgs-cluster-13-control-plane-
nb5r6                              1/1     Running     0            12d
kube-system                etcd-tkgs-cluster-13-control-plane-
zpcgs                              1/1     Running     0            12d
kube-system                kube-apiserver-tkgs-cluster-13-control-plane-
dpmjj                      1/1     Running     0            12d
kube-system                kube-apiserver-tkgs-cluster-13-control-plane-
nb5r6                      1/1     Running     0            12d
```

```
kube-system                    kube-apiserver-tkgs-cluster-13-control-plane-
zpcgs                          1/1     Running     0             12d
kube-system                    kube-controller-manager-tkgs-cluster-13-control-plane-
dpmjj                 1/1     Running     0             12d
kube-system                    kube-controller-manager-tkgs-cluster-13-control-plane-
nb5r6                 1/1     Running     1 (12d ago)   12d
kube-system                    kube-controller-manager-tkgs-cluster-13-control-plane-
zpcgs                 1/1     Running     0             12d
kube-system                    kube-
proxy-4kp57                                                        1/1     Running
0             12d
kube-system                    kube-
proxy-5q8pw                                                        1/1     Running
0             12d
kube-system                    kube-
proxy-5th6p                                                        1/1     Running
0             12d
kube-system                    kube-
proxy-8m6mx                                                        1/1     Running
0             12d
kube-system                    kube-proxy-
dn5lp                                                        1/1     Running
0             12d
kube-system                    kube-proxy-
qgmcg                                                        1/1     Running
0             12d
kube-system                    kube-proxy-
vbq27                                                        1/1     Running
0             12d
kube-system                    kube-proxy-
xhnws                                                        1/1     Running
0             12d
kube-system                    kube-proxy-
zgfvn                                                        1/1     Running
0             12d
kube-system                    kube-scheduler-tkgs-cluster-13-control-plane-
dpmjj                          1/1     Running     0             12d
kube-system                    kube-scheduler-tkgs-cluster-13-control-plane-
nb5r6                          1/1     Running     1 (12d ago)   12d
kube-system                    kube-scheduler-tkgs-cluster-13-control-plane-
zpcgs                          1/1     Running     0             12d
kube-system                    metrics-server-774bc4dc99-
qp7tb                                       1/1     Running     0             12d
vmware-system-auth             guest-cluster-auth-
svc-6m6cd                                         1/1     Running     0             12d
vmware-system-auth             guest-cluster-auth-svc-
h44xf                                         1/1     Running     0             12d
vmware-system-auth             guest-cluster-auth-svc-
l968n                                         1/1     Running     0             12d
vmware-system-cloud-provider   guest-cluster-cloud-provider-5f87d5d7d8-
rmd78                              1/1     Running     1 (12d ago)   12d
vmware-system-csi              vsphere-csi-controller-7d858778bd-
h7zhg                                   6/6     Running     4 (12d ago)   12d
vmware-system-csi              vsphere-csi-controller-7d858778bd-
rkl98                                   6/6     Running     0             12d
```

```
vmware-system-csi              vsphere-csi-controller-7d858778bd-
snmk7                                    6/6     Running     0             12d
vmware-system-csi              vsphere-csi-
node-22fnt                                                   3/3     Running     1 (12d
ago)   12d
vmware-system-csi              vsphere-csi-
node-5jtbr                                                   3/3     Running
0             12d
vmware-system-csi              vsphere-csi-
node-87lz6                                                   3/3     Running
0             12d
vmware-system-csi              vsphere-csi-node-
gp9sf                                                   3/3     Running     0
12d
vmware-system-csi              vsphere-csi-node-
k2psv                                                   3/3     Running     0
12d
vmware-system-csi              vsphere-csi-node-
mg8bw                                                   3/3     Running     0
12d
vmware-system-csi              vsphere-csi-node-
pctmv                                                   3/3     Running     0
12d
vmware-system-csi              vsphere-csi-node-
sslrl                                                   3/3     Running     1 (12d ago)
12d
vmware-system-csi              vsphere-csi-node-
zbqbq                                                   3/3     Running     0
12d
```

Obtenga y describa el estado del clúster de TKG.

```
kubectl get tkc <clustername>
```

```
kubectl describe tkc <clustername>
```

## Comprobar el estado del administrador de controladoras de TKG

Compruebe el estado del administrador de controladoras de TKG.

```
kubectl get deployments -n vmware-system-tkg vmware-system-tkg-controller-manager -o yaml
```

## Comprobar el estado del operador de máquinas virtuales

Los pods deben estar en ejecución.

```
kubectl get pods -n vmware-system-vmop
NAME                                                    READY   STATUS    RESTARTS   AGE
vmware-system-vmop-controller-manager-c8499b9df-5h6f9   2/2     Running   0          27d
vmware-system-vmop-controller-manager-c8499b9df-6wgr7   2/2     Running   0          27d
vmware-system-vmop-controller-manager-c8499b9df-tvbg6   2/2     Running   0          27d
vmware-system-vmop-hostvalidator-8498cc5f4d-vqhnk       1/1     Running   0          27d
```

El operador de máquina virtual crea `VirtualNetworkInterface` y comprueba su estado. Si una máquina virtual de nodo no obtiene una IP es la primera zona que habría que comprobar. ¿Ha pasado esta fase la creación de máquinas virtuales?

El operador de máquina virtual también es responsable de la conciliación de `VirtualMachineService` y la actualización de su estado. Si su IP externa no puede acceder a una API de Kubernetes de clústeres de TKG, compruebe el registro del operador de máquina virtual.

Por ejemplo, elija uno de los pods de operador de máquina virtual, especifique el espacio de nombres y especifique el contenedor del administrador. (El comando `logs` es para un contenedor. Dentro de cualquier pod de controladora hay un contenedor de administrador cuyos registros se pueden comprobar).

```
kubectl logs -f vmware-system-vmop-controller-manager-c8499b9df-5h6f9 -n vmware-system-vmop
manager
```

# Solucionar problemas de conexión del clúster de TKG y errores de inicio de sesión

Utilice esta sección para solucionar problemas de conexión del clúster de TKG y errores de inicio de sesión.

## Error de permisos insuficientes

Si no tiene suficientes permisos en espacio de nombres de vSphere, no puede conectarse a Supervisor ni a un clúster de TKG como usuario de vCenter Single Sign-On.

El complemento de vSphere para kubectl devuelve el mensaje de error `Error from server (Forbidden)` cuando intenta conectarse a una instancia de Supervisor o a un clúster de TKG como usuario de vCenter Single Sign-On.

No tiene permisos de función suficientes en el espacio de nombres de vSphere o no se otorgó acceso a su cuenta de usuario.

Si es un ingeniero de desarrollo y operaciones que trabaja con el clúster, compruebe con el administrador de vSphere que se le hayan concedido los permisos **Editar** de espacio de nombres de vSphere. Si es un desarrollador que utiliza el clúster para implementar cargas de trabajo, compruebe con el administrador de clústeres que se le haya concedido el acceso al clúster.

## Error de inicio de sesión de vSphere de Kubectl

Si recibe el siguiente error al intentar iniciar sesión en Supervisor o el clúster de TKG mediante el complemento de vSphere para kubectl, es posible que se deba a un error de inicio de sesión.

```
Failed to get available workloads, response from the server was invalid.
```

Para solucionar los errores de inicio de sesión, utilice `-v=10` para obtener una salida de registro más detallada.

```
kubectl vsphere login --server=10.110.150.56 --vsphere-username user@vsphere.local -v=10
```

Por ejemplo, a continuación se muestra el uso de salida detallada para mostrar un error de `invalid or missing credentials`.

```
DEBU[0000] User passed verbosity level: 10
DEBU[0000] Setting verbosity level: 10
DEBU[0000] Setting request timeout:
DEBU[0000] login called as: /usr/local/bin/kubectl-vsphere login --server=10.110.150.56 --
vsphere-username user@vpshere.local -v=10
DEBU[0000] Creating wcp.Client for --server=10.110.150.56.
INFO[0000] Does not appear to be a vCenter or ESXi address.
DEBU[0000] Got response:
INFO[0000] Using user@vpshere.local as username.
DEBU[0000] Env variable KUBECTL_VSPHERE_PASSWORD is present
DEBU[0000] Error while getting list of workloads: invalid or missing credentials
FATA[0000] Failed to get available workloads, response from the server was invalid.
```

## SSH para Supervisor

Es posible que sea necesario utilizar SSH para Supervisor para solucionar los errores de inicio de sesión.

**Advertencia** Cuando utiliza SSH en un nodo de plano de control de Supervisor, tiene permisos para dañar de forma permanente el clúster de Supervisor. Si el soporte de VMware encuentra pruebas de que un cliente ha realizado cambios en los componentes de Supervisor en un nodo de plano de control de Supervisor, el soporte de VMware puede marcar el clúster de Supervisor como no compatible y requerir que se vuelva a implementar la solución de vSphere IaaS control plane. Utilice esta sesión únicamente para probar redes, consultar registros y ejecutar los comandos logs/get/describe de kubectl. No implemente, elimine ni edite nada de esta sesión sin el permiso expreso de una base de conocimientos o el soporte de VMware.

Para utilizar SSH en un nodo de plano de control de Supervisor, realice los siguientes pasos.

1 Inicie sesión en vCenter con la cuenta de usuario raíz.

2 Escriba `dcli +i` para utilizar la CLI del centro de datos en modo interactivo.

3 Ejecute el comando `namespacemanagement software clusters list` para devolver el estado del Supervisor.

4 Escriba `exit` para salir del shell de `dcli`.

5 Escriba `shell` para entrar en el modo de shell de Bash.

6 Escriba `/usr/lib/vmware-wcp/decyptK8Pwd.py` para obtener la dirección IP y la contraseña de Supervisor.

7 Escriba `ssh 10.100.150.56` para ssh en Supervisor, donde debe reemplazar la dirección IP de ejemplo por la dirección IP devuelta por el comando anterior.

# Solucionar errores de la biblioteca de contenido

Consulte las sugerencias de este tema para corregir los errores de la biblioteca de contenido de versión de Tanzu Kubernetes.

## No se encontraron recursos de TKR

Un administrador de vSphere creó una biblioteca de contenido y la sincronizó con las versiones de Tanzu Kubernetes compatibles. Ha asignado la biblioteca de contenido al espacio de nombres de vSphere en el que va a implementar clústeres de TKG. Inició sesión en Supervisor y cambió el contexto a espacio de nombres de vSphere.

Al ejecutar los siguientes comandos, se muestra `No resources found`.

```
kubectl get tanzukubernetesreleases
```

```
kubectl get tkr
```

Para solucionar problemas, ejecute los siguientes comandos.

```
kubectl get virtualmachineimages -A
```

```
kubectl get vmimage -o wide
```

Compruebe si la biblioteca de contenido existe y está registrada con el espacio de nombres.

```
kubectl get contentsources
```

```
 kubectl get contentsourcebindings -A
```

Para resolverlo, inicie sesión en la interfaz de administración de vCenter Server. Desplácese hasta **Servicios**, seleccione el **Servicio de biblioteca de contenido** y haga clic en **Reiniciar**.

Si esto no resuelve el problema, es posible que deba eliminar la biblioteca de contenido del espacio de nombres. Para ello, cree una nueva biblioteca de contenido y agréguela al espacio de nombres, después de eliminar la anterior.

## Error al obtener elementos de la biblioteca

Al intentar aprovisionar un clúster de TKG, no se pueden extraer elementos de una biblioteca de contenido suscrita que está sincronizada y asociada con el espacio de nombres de vSphere.

Aparece el siguiente error:

```
Internal error occurred: get library items failed for.
```

Si la biblioteca de contenido suscrita alcanza los límites de capacidad de almacenamiento, no se pueden aprovisionar clústeres de TKG. La biblioteca de contenido está respaldada por el almacenamiento conectado. Con el paso del tiempo, a medida que se publiquen más versiones de Kubernetes y se sincronicen archivos OVA en la biblioteca, el almacenamiento podría llenarse hasta completar su capacidad.

Si va a sincronizar automáticamente las TKR, considere la posibilidad de cambiar a la sincronización manual y solo almacenar localmente las imágenes de TKR que necesite. Si ya utiliza la sincronización a petición, elimine de la biblioteca esas imágenes que ya no necesite. Como alternativa, puede migrarlas a una nueva biblioteca de contenido.

## No se puede encontrar TKR en la biblioteca de contenido local

Las bibliotecas de contenido locales se pueden utilizar en entornos restringidos de Internet.

Al crear la biblioteca de contenido local, tiene la opción de aplicar una directiva de seguridad a la biblioteca. Incluso si versión de Tanzu Kubernetes se cargó en la biblioteca, la versión no estará disponible para que la usen los clústeres de TKG si se da una de las siguientes condiciones.

- El paquete OVF de la biblioteca de contenido no está firmado.
- El paquete OVF está firmado con un certificado no válido.
- El paquete OVF está firmado con un certificado que no es de confianza para la instancia de vCenter Server donde está configurada la biblioteca de contenido local.

Cuando cargue los archivos OVA y VMDK en la biblioteca de contenido, asegúrese de que los archivos de manifiesto del certificado estén en el directorio de inicio desde donde está cargando el archivo.

# Corregir errores de clase de máquina virtual

Las clases de máquina virtual deben estar asociadas a la instancia de espacio de nombres de vSphere donde se aprovisionan los clústeres de servicio TKG.

## Error de enlace de clase de máquina virtual

Si intenta aprovisionar un clúster de TKGS mediante una o varias clases de máquina virtual que no agregó al espacio de nombres de vSphere de destino, se mostrará el error `VirtualMachineClassBindingNotFound`, cuyo ejemplo se muestra a continuación.

```
conditions:
  - lastTransitionTime: "2021-04-25T02:50:58Z"
    message: 1 of 2 completed
    reason: VirtualMachineClassBindingNotFound @ Machine/test-cluster
    severity: Error
    status: "False"
    type: ControlPlaneReady
  - lastTransitionTime: "2021-04-25T02:49:21Z"
    message: 0/1 Control Plane Node(s) healthy. 0/2 Worker Node(s) healthy
```

```
    reason: WaitingForNodesHealthy
    severity: Info
    status: "False"
    type: NodesHealthy
```

Para corregir el error, configure espacio de nombres de vSphere con las clases de máquina virtual que desea utilizar para el clúster de servicio TKG. Ejecute el comando `kubectl get virtualmachineclass` para ver las clases de máquinas virtuales asociadas con espacio de nombres de vSphere.

**Aviso** El comando `kubectl get virtualmachineclassbindings` quedó obsoleto a partir de vSphere 8 U3. El comando correcto que se debe utilizar es `virtualmachineclass`.

**Advertencia** El comando `kubectl get virtualmachineclasses` devuelve todas las clases de máquina virtual disponibles en Supervisor. Sin embargo, debido a que solo se pueden utilizar las clases de máquina virtual que están asociadas con la instancia de espacio de nombres de vSphere de destino para aprovisionar un clúster, la versión plural del nombre es de solo información y no se puede confiar en ella al aprovisionar.

# Solucionar errores de aprovisionamiento de clústeres de TKGS

Si no puede aprovisionar un clúster de TKGS, revise esta lista de errores comunes para solucionar problemas.

## Comprobar registros de la API del clúster

Si no puede crear un clúster de TKG, compruebe que CAPW/V funcione.

El controlador CAPW/V es la implementación específica de la infraestructura de la API del clúster. CAPW/V se habilita a través de Supervisor. CAPW/V es un componente de TKG y es responsable de administrar el ciclo de vida de los clústeres de TKG.

CAPW/V es responsable de crear y actualizar la red virtual. La creación de los nodos del clúster solo puede avanzar si la red virtual está lista. ¿El flujo de trabajo de creación del clúster pasó por esta fase?

CAPW/V es responsable de crear y actualizar el servicio de máquina virtual. ¿Se creó correctamente el servicio de máquina virtual? ¿Obtuvo la dirección IP externa? ¿El flujo de trabajo de creación del clúster pasó por esta fase?

Para responder estas preguntas, compruebe el registro de la API del clúster de la siguiente manera:

```
kubectl config use-context tkg-cluster-ns
```

```
kubectl get pods -n vmware-system-capw  | grep capv-controller
```

```
kubectl logs -n vmware-system-capw -c manager capv-controller-manager-...
```

## Error de validación de la especificación del clúster

De acuerdo con la especificación de YAML, se permite el uso del carácter de espacio en un nombre de clave. Es una cadena escalar que contiene un espacio y no requiere comillas.

Sin embargo, la validación de TKGS no permite el uso del carácter de espacio en los nombres de clave. En TKGS, un nombre de clave válido solo debe constar de caracteres alfanuméricos, un guion (como `key-name`), un guion bajo (como `KEY_NAME`) o un punto (como `key.name`).

Si utiliza el carácter de espacio en un nombre de clave en la especificación del clúster, el clúster de TKGS no se implementa. El registro vmware-system-tkg-controller-manager muestra el siguiente mensaje de error:

```
Invalid value: \"Key Name\": a valid config key must consist of alphanumeric
characters, '-', '_' or '.' (e.g. 'key.name', or 'KEY_NAME', or 'key-name', regex used
for validation is '[-._a-zA-Z0-9]+')"
```

Para solucionar el error, elimine el carácter de espacio por completo o reemplácelo por uno compatible.

## Errores al aplicar el YAML del clúster de TKG

Si recibe mensajes de error al aplicar el YAML del clúster de TKG, solucione los problemas de la siguiente manera.

**La red del clúster no está en el estado correcto**

Comprender el flujo de trabajo de aprovisionamiento de clústeres de TKG:

- CAPV crea un objeto de red virtual para cada red del clúster de TKG.
- Si Supervisor está configurado con redes NSX, NCP detecta los objetos de red virtual y crea un enrutador de nivel 1 de NSX y un segmento de NSX para cada red virtual.
- CAPV comprueba el estado de la red virtual y, cuando está lista, continúa con el paso siguiente del flujo de trabajo.

La controladora de servicio de máquina virtual observa los objetos personalizados creados por CAPV y utiliza esas especificaciones para crear y configurar las máquinas virtuales que conforman el clúster de TKG.

NSX Container Plugin (NCP) es una controladora que detecta los recursos de red agregados a etcd a través de la API de Kubernetes y organiza la creación de los objetos correspondientes en NSX.

Todas estas controladoras se ejecutan como pods de Kubernetes en el plano de control de Supervisor. Para solucionar problemas de red, compruebe el registro de la controladora CAPV, el registro del servicio de máquina virtual y el registro de NCP.

Compruebe los registros del contenedor, en los que `name-XXXX` es el nombre único del pod cuando ejecuta lo siguiente:

```
kubectl get pods -A
kubectl logs pod/name-XXXXX -c pod-name -n namesapce
```

### Recuento de nodos de plano de control no válido

El clúster de TKG en Supervisor admite 1 o 3 nodos de plano de control. Si introduce otra cantidad de réplicas, se produce un error en el aprovisionamiento del clúster.

### Clase de almacenamiento no válida para la máquina virtual de plano de control o de trabajo

Ejecute el siguiente comando:

```
kubectl describe ns <tkg-clsuter-namesapce>
```

Asegúrese de que se haya asignado una clase de almacenamiento al espacio de nombres en el que intenta crear el clúster de TKG. Debe haber una cuota de recursos en el espacio de nombres de vSphere que haga referencia a esa clase de almacenamiento, la cual debe existir en Supervisor.

Asegúrese de que el nombre coincida con la clase de almacenamiento presente en Supervisor. Ejecute `kubectl get storageclasses` Supervisor como administrador de vSphere. Es posible que WCP modifique el nombre al aplicar el perfil de almacenamiento en Supervisor (por ejemplo, los guiones se convierten en guiones bajos).

### Clase de máquina virtual no válida

Asegúrese de que el valor proporcionado en el YAML del clúster coincida con una de las clases de máquina virtual que devuelve `kubectl get virtualmachineclass`. Un clúster de TKG solo puede utilizar clases enlazadas. Una clase de máquina virtual enlazada es aquella que se agregó al espacio de nombres de vSphere.

El comando `kubectl get virtualmachineclasses` devuelve todas las clases de máquina virtual en Supervisor, pero solo se pueden utilizar las que están enlazadas.

### No se pueden encontrar las distribuciones TKR

Si aparece un error similar al siguiente:

```
"Error from server (unable to find Kubernetes distributions):
admission webhook "version.mutating.tanzukubernetescluster.run.tanzu.vmware.com"
denied the request: unable to find Kubernetes distributions"
```

Probablemente se trate de un problema de la biblioteca de contenido. Para enumerar lo que está disponible, utilice el comando `kubectl get virtualmachineimages -A`. El resultado es lo que está disponible, sincronizado o cargado en la biblioteca de contenido.

Para TKG en Supervisor, hay nuevos nombres de TKR que son compatibles con la nueva API de TKR. Debe asegurarse de asignar el nombre correcto a cada TKR en la biblioteca de contenido.

Nombre en la biblioteca de contenido: `photon-3-amd64-vmi-k8s-v1.23.8---vmware.2-tkg.1-zshippable`

Nombre correspondiente en la especificación del clúster de TKG: `version: v1.23.8+vmware.2-tkg.1-zshippable`

## Se aplica el YAML de TKG, pero no se crean máquinas virtuales

Si el YAML del clúster de TKG 2.0 es válido y se aplica, pero no se crean las máquinas virtuales del nodo, solucione el problema de la siguiente manera.

### Comprobar los recursos CAPI/CAPV

Comprobar si TKG creó los recursos de nivel CAPI/CAPV.

- Compruebe si CAPV creó los recursos de máquina virtual.
- Consulte los registros de operador de máquina virtual para ver por qué no se creó la máquina virtual; por ejemplo, es posible que se haya producido un error en la implementación de OVF debido a la falta de recursos en el host ESX.
- Compruebe los registros de operador de máquina virtual y CAPV.
- Compruebe los registros de NCP. NCP es responsable de obtener la red virtual, la interfaz de la red virtual y el equilibrador de carga para el plano de control. Si se produce algún error relacionado con esos recursos, puede ser un problema.

### Error de servicios de máquina virtual

Error de servicios de máquina virtual

- Ejecute `kubectl get virtualmachineservices` en el espacio de nombres.
- ¿Se creó un servicio de máquina virtual?
- Ejecute `kubectl describe virtualmachineservices` en el espacio de nombres.
- ¿Hay errores notificados en el servicio de máquina virtual?

### Error de red virtual

Ejecute `kubectl get virtualnetwork` en el espacio de nombres.

¿Se creó la red virtual para este clúster?

Ejecute `kubectl describe virtual network` en el espacio de nombres.

¿Se creó la interfaz de red virtual para la máquina virtual?

## El plano de control del clúster de TKG no se está ejecutando

Si el plano de control de TKG no se está ejecutando, compruebe si los recursos estaban listos cuando se produjo el error. ¿Lo que no está activo es el plano de control de un nodo de unión o un nodo de inicialización? Además, compruebe si el identificador del proveedor no está establecido en el objeto de nodo.

**Comprobar si los recursos estaban listos cuando se produjo el error**

Además de la búsqueda de registros, la comprobación del estado "Equilibrador de carga del plano de control" de los objetos relacionados lo ayudará a comprender si los recursos estaban listos cuando se produjo el error. Consulte la solución de problemas de red.

**¿Lo que no está activo es el plano de control de un nodo de unión o un nodo de inicialización?**

Las uniones de nodos a veces no funcionan correctamente. Consulte los registros de nodo de una determinada máquina virtual. Es posible que al clúster le falten nodos de trabajo y de plano de control si el nodo de inicialización no se activa correctamente.

**No está establecido el identificador del proveedor en el objeto de nodo**

Si se creó la máquina virtual, compruebe si tiene direcciones IP y, a continuación, consulte los registros de cloud-init (los comandos kubeadm se ejecutan correctamente).

Consulte los registros de la controladora CAPI para verificar si hay algún problema. Para comprobarlo, utilice los `kubectl get nodes` en el clúster de TKG y, a continuación, verifique si existe el identificador del proveedor en el objeto de nodo.

## No se crean nodos de trabajo de TKG

Si se crean el clúster de TKG y las máquinas virtuales del plano de control, pero no se crean trabajos ni otros objetos de máquina virtual, pruebe lo siguiente:

```
kubectl describe cluster CLUSTER-NAME
```

Compruebe si hay recursos de máquina virtual en el espacio de nombres. ¿Se crearon otros?

Si no es así, consulte los registros de CAPV para ver por qué no se crean los otros datos de arranque de objetos de máquina virtual que no están disponibles.

Si CAPI no puede comunicarse con el plano de control del clúster de TKG a través del equilibrador de carga, ya sea NSX con la IP de máquina virtual del nodo o VDS con el equilibrador de carga externo, obtenga el kubeconfig del clúster de TKG mediante el secreto en el espacio de nombres:

Utilice el secreto en el espacio de nombres para obtener el archivo kubeconfig del clúster de TKG:

```
kubectl get secret -n <namespace> <tkg-cluster-name>-kubeconfig -o jsonpath='{.data.value}' |
base64 -d
> tkg-cluster-kubeconfig; kubectl --kubeconfig tkg-cluster-kubeconfig get pods -A
```

Si aparece el error de conexión rechazada, es probable que el plano de control no se haya inicializado correctamente. Si se agota el tiempo de espera de E/S, compruebe la conectividad con la dirección IP en el archivo kubeconfig.

NSX con equilibrador de carga integrado:

- Compruebe que el equilibrador de carga de plano de control esté activo y que se pueda acceder a él.
- Si el equilibrador de carga no tiene IP, consulte los registros de NCP y la interfaz de usuario de NSX-T para ver si los componentes relacionados están en los estados correctos. (El equilibrador de carga de NSX-T, el servidor virtual y el grupo de servidores deben estar en el estado correcto).
- Si el equilibrador de carga tiene IP, pero no se puede acceder a él (`curl -k https://<LB-VIP>:6443/healthz` debería devolver un error no autorizado).

  Si el tipo de equilibrador de carga de la IP externa del servicio se encuentra en estado 'pendiente', compruebe que el clúster de TKG pueda comunicarse con la API del supervisor de Kubernetes a través de la VIP del LB del supervisor. Asegúrese de que no haya ninguna superposición de direcciones IP.

Comprobar si los nodos del plano de control de TKG están en buen estado:

- Compruebe si el plano de control del clúster de TKG informa de algún error (por ejemplo, no se puede crear el nodo con el identificador del proveedor).
- El proveedor de nube del clúster de TKG no marcó el nodo con el identificador del proveedor correcto, por lo que CAPI no puede comparar el identificador del proveedor en el nodo del clúster invitado con el recurso de máquina en el clúster supervisor para hacer la comprobación.

Acceda mediante SSH a la máquina virtual de plano de control o utilice el archivo kubeconfig del clúster de TKG para comprobar si el pod del proveedor de nube de TKG se está ejecutando o registra errores. Consulte Conectarse a clústeres de Servicio TKG como usuario del sistema y administrador de Kubernetes.

```
kubectl get po -n vmware-system-cloud-provider
```

```
kubectl logs -n vmware-system-cloud-provider <pod name>
```

Si VMOP no concilió VirtualMachineService correctamente, compruebe el registro del operador de máquinas virtuales.

Si NCP tenía problemas para crear recursos de NSX-T, compruebe el registro de NCP.

Si el plano de control no se inicializó correctamente, determine la IP de la máquina virtual. El estado debe contener la IP de la máquina virtual.

```
kubectl get virtualmachine -n <namespace> <TKC-name>-control-plane-0 -o yaml
```

```
ssh vmware-system-user@<vm-ip> -i tkc-cluster-ssh
```

Compruebe si kubeadm registró algún error.

```
cat /var/log/cloud-init-output.log | less
```

## El clúster de TKG aprovisionado se detuvo en la fase "Crear"

Ejecute los siguientes comandos para comprobar el estado del clúster.

```
kubectl get tkc -n <namespace>
```

```
kubectl get cluster -n <namespace>
```

```
kubectl get machines -n <namespace>
```

KubeadmConfig estaba presente, pero CAPI no pudo encontrarlo. Se comprobó si el token de vmware-system-capv tenía los permisos adecuados para consultar kubeadmconfig.

```
$kubectl --token=__TOKEN__ auth can-i get kubeadmconfig
yes
```

Es posible que la memoria caché de tiempo de ejecución de la controladora no se estaba actualizando. Es probable que las memorias caché del reloj CAPI estén obsoletas y no detecten los nuevos objetos. Si es necesario, reinicie capi-controller-manager para resolver el problema.

```
kubectl rollout restart deployment capi-controller-manager -n vmware-system-capv
```

## espacio de nombres de vSphere se detuvo en la fase "Finalización"

Compruebe que TKR, Supervisor y vCenter estén sincronizados desde una perspectiva de compatibilidad de versión.

Los espacios de nombres solo se pueden eliminar cuando, a su vez, se eliminan todos los recursos dentro de ellos.

```
kubectl describe namespace NAME
```

Se encontró el siguiente error: "Error del servidor (no se pueden encontrar las distribuciones de Kubernetes): webhook de admisión "version.mutating.tanzukubernetescluster.run.tanzu.vmware.com" rechazó la solicitud: no se pueden encontrar las distribuciones de Kubernetes".

Compruebe las imágenes de máquina virtual en vCenter.

```
kubectl get virtualmachineimages -A
```

# Solucionar errores de nodo del clúster de servicio TKG

Si aprovisionó un clúster de TKGS y se crearon los nodos, pero una o varias máquinas virtuales no se inician o tienen errores, siga estos consejos para solucionarlo.

## Comprobar CRD relevantes

Una vez que se crean las máquinas virtuales, se deben crear las CRD correspondientes. Compruebe si se crearon y existen las CRD relevantes (implementaciones de máquinas y máquinas virtuales).

Comprobar implementaciones de máquinas:

```
kubectl get machinedeployments -A -o wide
```

Comprobar máquinas virtuales:

```
kubectl get virtualmachines -A
```

## Comprobar tamaño de nodo

Aprovisionó un clúster de TKG. El sistema está intentando encender las máquinas virtuales del plano de control, pero se muestra el siguiente mensaje de error.

```
The host does not have sufficient CPU resources to satisfy the reservation.
```

El tamaño o la clase de la máquina virtual no son suficientes para la implementación del clúster. Cambie el tipo o la clase de la máquina virtual. Evite usar los tipos de clase de máquina virtual extrapequeña y pequeña tanto para el plano de control como para los nodos de trabajo.

# Solucionar errores de redes del clúster de servicio TKG

Consulte los consejos de esta sección para corregir los errores de redes de clústeres de TKGS.

## Comprobar redes de nodos

Cada clúster de TKG debe tener los siguientes recursos de red.

| Objeto de red | Recursos de red | Descripción | Solucionar problemas | Comando |
|---|---|---|---|---|
| Red virtual | Enrutador de nivel 1 y segmento vinculado | Red de nodo para el clúster | Asegurarse de que la IP de SNAT esté asignada | `kubectl get virtualnetwork -n NS-NAME` |
| VirtualNetworkInterface | Puerto lógico en el segmento | Interfaz de red de nodo para nodos del clúster | Asegurarse de que cada máquina virtual tenga una dirección IP | `kubectl get virtualmachines -n NS-NAME NODE-NAME` |

## Comprobar el equilibrador de carga para el plano de control

El equilibrador de carga del plano de control del clúster de TKG proporciona acceso al servidor de la API de Kubernetes. El sistema aprovisiona automáticamente este equilibrador de carga durante la creación del clúster. Debe tener los siguientes recursos.

Comprobar el estado del equilibrador de carga del plano de control puede ayudarlo a comprender si los recursos estaban listos cuando se produjeron errores. En general, puede encontrar este equilibrador de carga mediante Buscar estos equilibradores de carga con este comando en el clúster supervisor: `kubectl get services -A | grep control-plane-service`

| Objeto de red | Recursos de red | Descripción | Solucionar problemas | Comando |
|---|---|---|---|---|
| Servicio de máquina virtual | N/C | Se crea el servicio de máquina virtual y se traduce a un servicio k8s. | Asegúrese de que su estado esté actualizado y que incluya la IP virtual (VIP) del equilibrador de carga. | `kubectl get virtualmachineservices -n NS-NAME SERVICE-NAME` |
| Servicio | Servidor de equilibrador de carga con instancia de servidor virtual y grupo de servidores asociado (grupo de miembros) | Se crea un servicio de Kubernetes de tipo equilibrador de carga para acceder al servidor de API del clúster de TKG. | Asegúrese de que se haya asignado una IP externa. Asegúrese de que puede acceder a la API del clúster de TKG a través de la IP externa del servicio de equilibrador de carga. | Espacio de nombres de supervisor:<br>`kubectl get services -A \| grep control-plane-service`<br>Espacio de nombres del clúster:<br>`kubectl get services -n NS-NAME`<br>Cualquier espacio de nombres<br>`curl -k https://EXTERNAL-IP:PORT/healthz` |
| Endpoints | Los miembros del endpoint (nodos de plano de control del clúster de TKG) deben estar en el grupo de miembros. | Se crea un endpoint para incluir todos los nodos del plano de control del clúster de TKG. | | `kubectl get endpoints -n NS-NAME SERVICE-NAME` |

## Comprobar los servicios del equilibrador de carga en los nodos de trabajo

El usuario crea una instancia de equilibrador de carga para los nodos de trabajo del clúster de TKG cuando se crea un servicio de Kubernetes de tipo equilibrador de carga.

El primer paso es asegurarse de que el proveedor de nube se esté ejecutando en el clúster de TKG.

```
kubectl get pods -n vmware-system-cloud-provider
```

Compruebe que se hayan creado los objetos de Kubernetes relacionados y que tengan un estado correcto.

| Objetos de red | Recursos de red | Descripción | Comando |
| --- | --- | --- | --- |
| Servicio de máquina virtual en supervisor | N/C | Se crea un servicio de máquina virtual en supervisor y se traduce en un servicio de Kubernetes en supervisor | `kubectl get virtualmachineservice -n NS-NAME SVC-NAME` |
| Servicio de equilibrador de carga en supervisor | Servidor virtual en el equilibrador de carga del clúster de TKG y un grupo de miembros asociado. | Se crea el servicio de equilibrador de carga en supervisor para acceder a este tipo de servicio de equilibrador de carga | `kubectl get services -n NS-NAME SVC-NAME` |
| Endpoints en Supervisor | Los miembros del endpoint (nodos de trabajo del clúster de TKG) deben estar en el grupo de miembros en NSX. | Se crea un endpoint para incluir todos los nodos de trabajo del clúster de TKG | `# kubectl get endpoints -n NS-NAME SVC-NAME` |
| Servicio de equilibrador de carga en el clúster de TKG | N/C | El servicio del equilibrador de carga en el clúster de TKG implementado por el usuario debe tener su estado actualizado con la IP del equilibrador de carga | `kubectl get services` |

## Comprobar la pila de redes NSX Supervisor

El servidor de API de Kubernetes, el pod de NCP y el contenedor del administrador que se ejecuta en cualquier pod de controlador son los puntos de partida principales para comprobar los problemas de redes de infraestructura.

El siguiente mensaje de error puede indicar un problema de enrutamiento o MTU en cualquier punto del tejido de red, incluido el grupo de puertos físicos al que están conectadas las NIC de hosts ESXi:

```
{"log":"I0126 19:40:15.347154 1 log.go:172] http: TLS handshake error from
100.64.128.1:4102: EOF\n","stream":"stderr","time":"2021-01-26T19:40:15.347256146Z"}
```

Para solucionar problemas, acceda mediante SSH al host ESXi y ejecute el siguiente comando:

```
esxcli network ip interface ipv4 get
```

Este comando enumera todas las interfaces de VMkernel del host. Si tiene una única interfaz de TEP, siempre será vmk10. Si tiene una interfaz de TEP de segundo o tercer nivel, será vmk11, vmk12 y así sucesivamente. La cantidad de interfaces de TEP que se crean depende de cuántos vínculos superiores haya asignado al TEP en el perfil de vínculo superior. Se crea una interfaz de TEP por vínculo superior si seleccionó "uso compartido de carga" para los TEP entre vínculos superiores.

El comando ping principal de TEP a TEP tiene la siguiente sintaxis:

```
vmkping ++netstack=vxlan -s 1572 -d -I vmk10 10.218.60.66
```

Donde

- `-s` es el tamaño del paquete
- `-d` significa no fragmentar
- `-I` significa que el origen del vínculo es vmk10
- `IP address` es una interfaz de TEP en otro host ESXi o NSX Edge al que está haciendo ping

Si la MTU se establece en 1600, se debe producir un error en un tamaño de paquete superior a 1573 (solo se necesitan MTU superiores a 1500). Si la MTU se establece en 1500, se debe producir un error en un valor superior a 1473. Es posible que desee cambiar esta opción a vmk11 si tiene interfaces de TEP adicionales desde las que desea obtener el ping.

# Reiniciar una actualización de clúster de TKG con errores

Si se produce un error en la actualización de un clúster de TKG, puede reiniciar el trabajo de actualización y volver a intentarlo.

### Problema

Se produce un error en la actualización de un clúster de TKG y, como consecuencia, el estado del clúster es `upgradefailed`.

### Causa

Se pueden producir errores en la actualización de un clúster por varios motivos, uno de los cuales puede ser un almacenamiento insuficiente. Para reiniciar un trabajo de actualización con errores y volver a intentar la actualización del clúster, complete el siguiente procedimiento.

### Solución

1 Inicie sesión en Supervisor como administrador.

2 Busque `update_job_name`.

   ```
   kubectl get jobs -n vmware-system-tkg -l "run.tanzu.vmware.com/cluster-namespace=$
   {cluster_namespace},cluster.x-k8s.io/cluster-name=${cluster_name}"
   ```

3 Ejecute `kubectl proxy` de modo que `curl` se pueda utilizar para enviar solicitudes.

   ```
   kubectl proxy &
   ```

   Debería ver `Starting to serve on 127.0.0.1:8001`.

   **Nota** No puede utilizar `kubectl` para revisar o actualizar el `.status` de un recurso.

4 Con `curl`, emita el siguiente comando de revisión para aumentar el valor de `.spec.backoffLimit`.

```
curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "replace", "path": "/spec/backoffLimit", "value": 8}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}
```

5 Con `curl`, emita el siguiente comando de revisión para borrar `.status.conditions` de modo que la controladora de trabajo pueda crear nuevos pods.

```
$ curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "remove", "path": "/status/conditions"}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}/
status
```

# Corregir errores de implementación de contenedores

Se pueden producir errores de implementación de contenedores si la directiva de seguridad de pods y el control de acceso basado en funciones no están configurados para los usuarios autenticados.

### Problema

La carga de trabajo de un contenedor se implementa en un clúster de TKG 2.0, pero la carga de trabajo no se inicia. Aparece un error similar al siguiente:

```
Error: container has runAsNonRoot and image will run as root.
```

### Causa

Los clústeres de TKG se aprovisionan con el controlador de admisión de PodSecurityPolicy habilitado. Ningún usuario autenticado puede crear pods con privilegios o sin privilegios hasta que el administrador de clústeres enlace PodSecurityPolicy a los usuarios autenticados.

### Solución

Si utiliza TKR 1.24 o una versión anterior, cree un enlace adecuado para la instancia de PodSecurityPolicy predeterminada o defina una instancia personalizada. Si utiliza TKR 1.25 o una versión posterior, configure la admisión de seguridad de pods. Consulte Capítulo 18 Administrar la seguridad para clústeres de servicio TKG.

# Solucionar errores del registro del contenedor

Puede utilizar un registro de contenedor externo con pods de clúster de TKG.

## Solucionar errores al extraer imágenes de un registro de contenedor

Si configura TKG con los certificados en los que confiar y agrega el certificado autofirmado al clúster kubeconfig, debería poder extraer correctamente una imagen de contenedor de un registro privado que use ese certificado autofirmado.

El siguiente comando puede ayudarle a determinar si la imagen del contenedor se extrajo correctamente para una carga de trabajo del pod:

```
kubectl describe pod PODNAME
```

Este comando muestra el estado detallado y los mensajes de error para un pod determinado. Un ejemplo de intento de extracción de una imagen antes de agregar certificados personalizados al clúster:

```

Events:
  Type     Reason                        Age                From               Message
  ----     ------                        ----               ----               -------
  Normal   Scheduled                     33s                default-scheduler  ...
  Normal   Image                         32s                image-controller   ...
  Normal   Image                         15s                image-controller   ...
  Normal   SuccessfulRealizeNSXResource  7s (x4 over 31s)  nsx-container-ncp  ...
  Normal   Pulling                       7s                 kubelet            Waiting test-gc-
e2e-demo-ns/testimage-8862e32f68d66f727d1baf13f7eddef5a5e64bbd-v10612
  Warning  Failed                        4s                 kubelet            failed to get
images: ... Error: ... x509: certificate signed by unknown authority
```

Y, al ejecutar el siguiente comando:

```
kubectl get pods
```

El error `ErrImagePull` también se puede ver en la vista de estado general del pod:

```

NAME                                         READY   STATUS         RESTARTS   AGE
testimage-nginx-deployment-89d4fcff8-2d9pz   0/1     Pending        0          17s
testimage-nginx-deployment-89d4fcff8-7kp9d   0/1     ErrImagePull   0          79s
testimage-nginx-deployment-89d4fcff8-7mpkj   0/1     Pending        0          21s
testimage-nginx-deployment-89d4fcff8-fszth   0/1     ErrImagePull   0          50s
testimage-nginx-deployment-89d4fcff8-sjnjw   0/1     ErrImagePull   0          48s
testimage-nginx-deployment-89d4fcff8-xr5kg   0/1     ErrImagePull   0          79s
```

Los errores “x509: certificado firmado por entidad desconocida” y “ErrImagePull” indican que el clúster no está configurado con el certificado correcto para conectarse al registro de contenedor privado. Falta el certificado o está mal configurado.

Si experimenta errores al conectarse a un registro privado después de configurar los certificados, puede comprobar si los certificados aplicados en la configuración se aplican al clúster. Puede comprobar si los certificados se aplicaron correctamente en su configuración mediante SSH.

Se pueden realizar dos pasos de investigación conectándose a un nodo de trabajador a través de SSH.

1 Compruebe la carpeta /etc/ssl/certs/ en busca de archivos denominados tkg-<cert_name>.pem, donde <cert_name> es la propiedad "name" del certificado agregado en TkgServiceConfiguration. Si los certificados coinciden con lo que está presente en TkgServiceConfiguration, y sigue sin poder usarse un registro privado, complete el siguiente paso para profundizar en el diagnóstico.

2 Ejecute la siguiente prueba de conexión openssl en el servidor de destino mediante certificados autofirmados con el comando openssl s_client -connect hostname:port_num, donde hostname es el nombre de host o el nombre de DNS del registro privado que utiliza certificados autofirmados, y port_num es el número de puerto en el que se ejecuta el servicio (por lo general, el puerto 443 para HTTPS).

   Puede comprobar exactamente qué error devuelve openssl cuando intenta conectarse al endpoint que utiliza certificados autofirmados y solucionar la situación desde allí; por ejemplo, agregando los certificados correctos a TkgServiceConfiguration. Si el clúster de TKG está integrado con el certificado incorrecto, deberá actualizar la configuración de TKG con los certificados correctos, eliminar el clúster de TKG y, a continuación, volver a crearlo con la configuración que contiene los certificados correctos.

3 Compruebe que el contenido del mapa de datos del secreto tenga doble codificación base64. Se requiere doble codificación base64. Si el contenido del valor de la asignación de datos no está doblemente codificado con Base64, no se puede procesar el archivo PEM resultante.

# Solucionar errores con certificados de CA de confianza adicionales

Consulte este tema si experimenta un problema al agregar certificados de CA de confianza adicionales a un clúster de TKG.

## Solucionar errores con certificados de CA de confianza adicionales

Mediante la API v1alpha3 o la API v1beta1, puede incluir una variable trust que contenga valores para certificados de CA de confianza adicionales. El caso práctico común es agregar un certificado de registro de contenedor privado al clúster. Consulte Integrar clústeres de Servicio TKG con un registro de contenedor privado.

Por ejemplo, use la API v1beta1:

```
topology:
  variables:
    - name: trust
      value:
        additionalTrustedCAs:
          - name: my-ca
```

Con el secreto de la siguiente manera:

```
apiVersion: v1
data:
  my-ca: # Double Base64 encoded CA certificate
kind: Secret
metadata:
  name: CLUSTER_NAME-user-trusted-ca-secret
  namespace: tap
type: Opaque
```

Cuando se agrega una instancia `trust.additionalTrustedCAs` a una especificación de clúster de TKG, Supervisor actualiza los nodos del clúster en una Información sobre el modelo de actualización gradual para clústeres de Servicio TKG. Sin embargo, si se produce un error en los valores de `trust`, las máquinas no aparecerán correctamente y no se unirán al clúster.

Si utiliza la API v1beta1, el contenido del certificado debe tener codificación base64 doble. Si el contenido del certificado no está codificado en base64 doble, es posible que aparezca el siguiente error.

```
ls cannot access '/var/tmp/_var_ib_containerd': No such file or directory
```

Si utiliza la API v1alpha3 (o la API v1alpha2), el contenido del certificado debe estar codificado en base64 único. Si el contenido del certificado no está codificado en base64, es posible que aparezca el siguiente error.

```
"default.validating.tanzukubernetescluster.run.tanzu.vmware.com" denied the request:
Invalid certificate internalharbor, Error decoding PEM block for internalharbor in the
TanzuKubernetesCluster spec's trust configuration
```

Si no utiliza la codificación adecuada, los nodos de máquina no aparecerán y recibirá el error anterior. Para solucionar el problema, codifique correctamente el contenido del certificado y agregue ese valor a la asignación de datos del secreto.

Puede ejecutar `kubectl replace -f /tmp/kubectl-edit-2005376329.yaml` para volver a probar esta actualización.